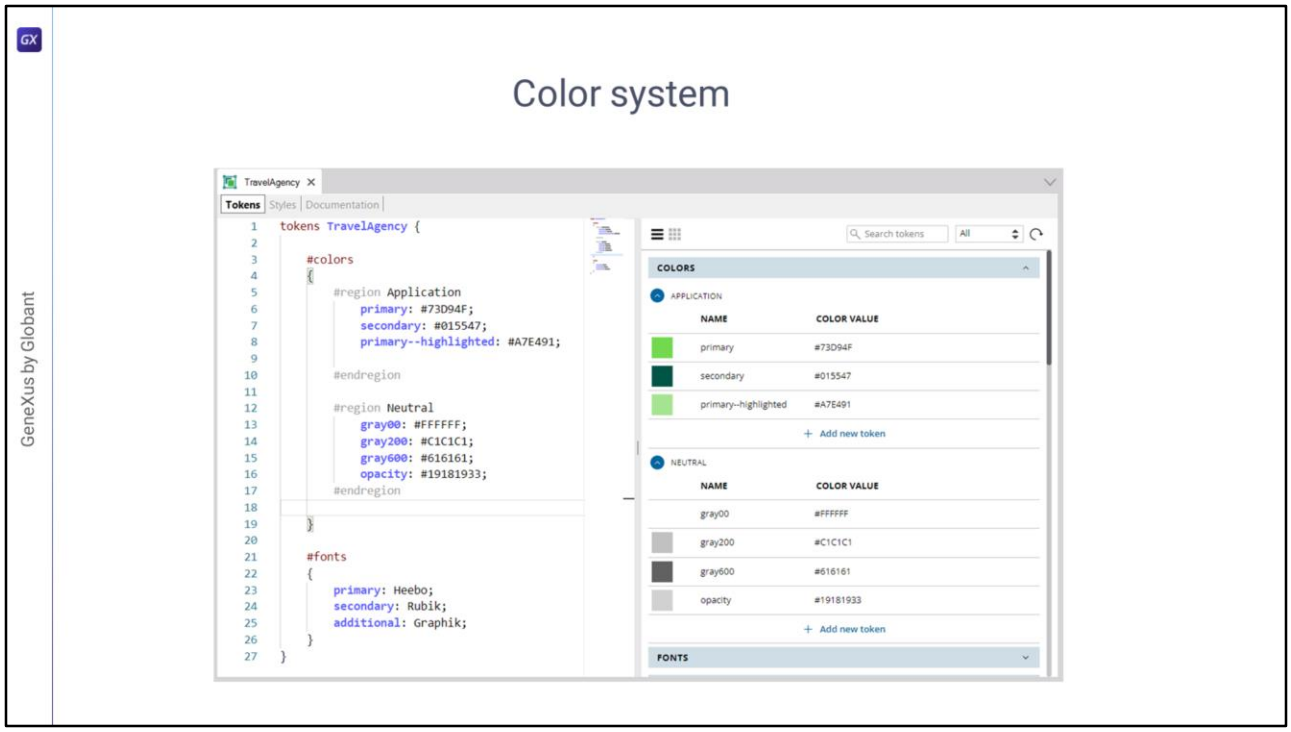
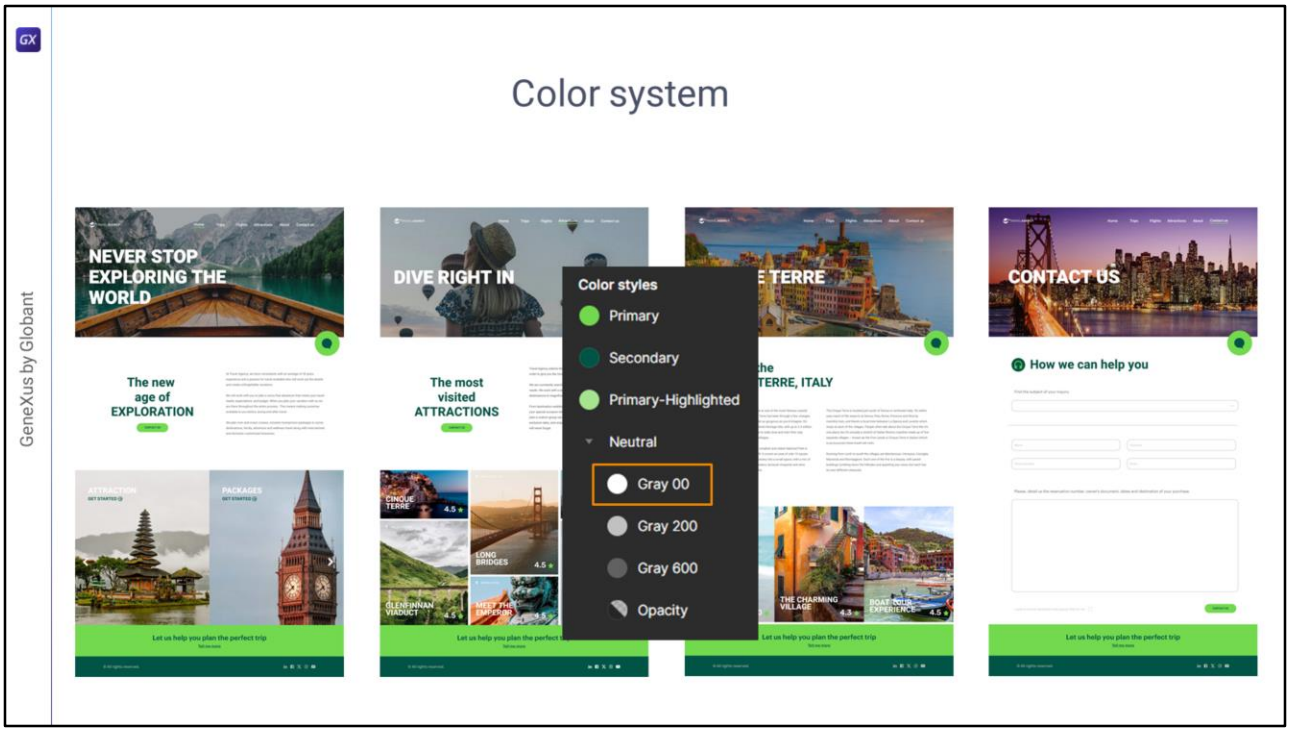
GeneXus by Globant

Design System’s color system

Cecilia Fernández

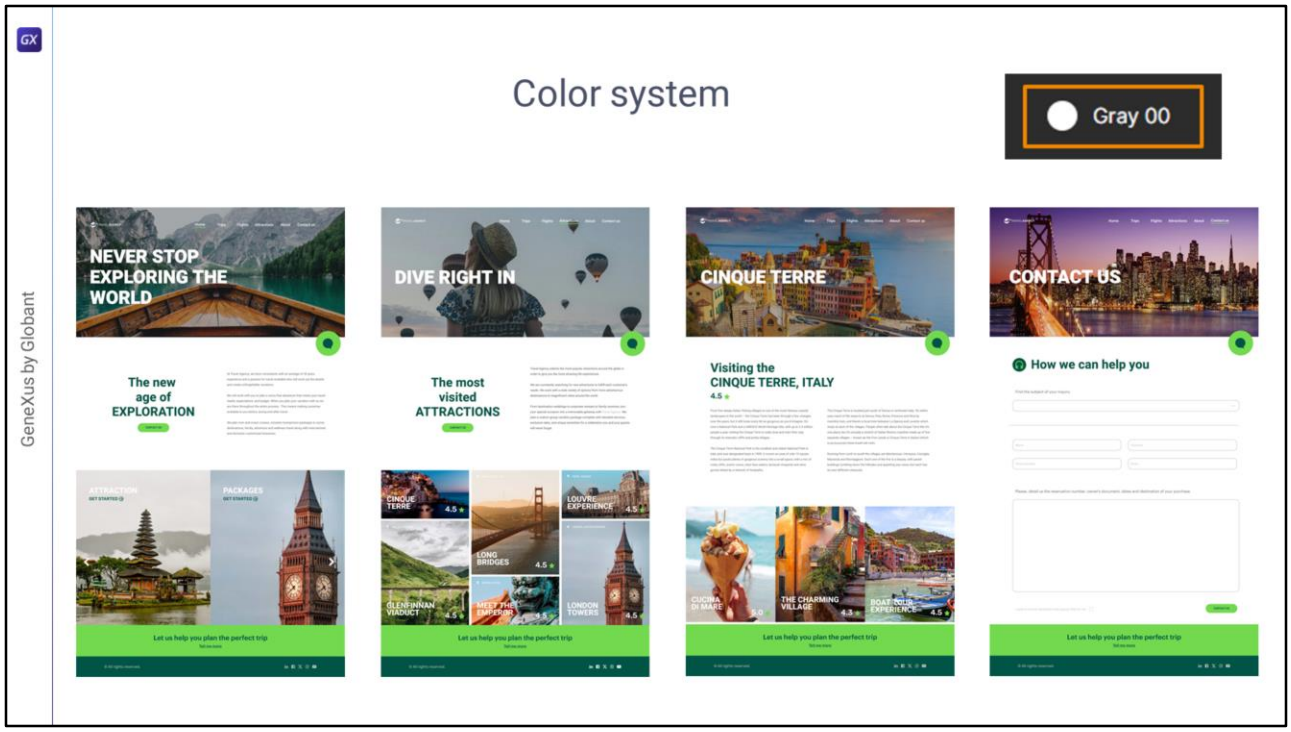


In the previous video, we had completed the color tokens of our DSO from what we had taken from the project in Figma, remember?

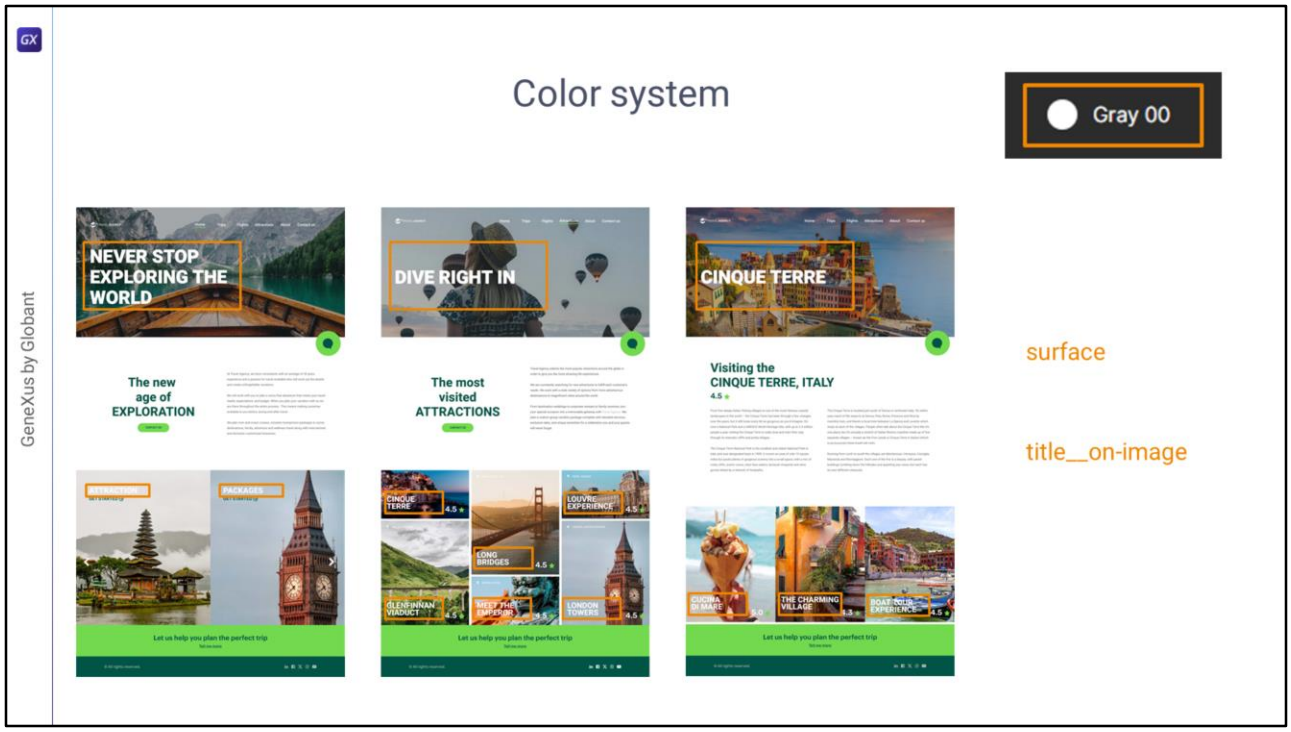


However, the color system expressed in this way is very basic: at the semantic level it only expresses the **primary** color of the application and the **secondary** color, but it does not express things such as what color to apply for the background, what color to apply for texts that are titles on backgrounds, for common texts on backgrounds, for texts on images, and so on.

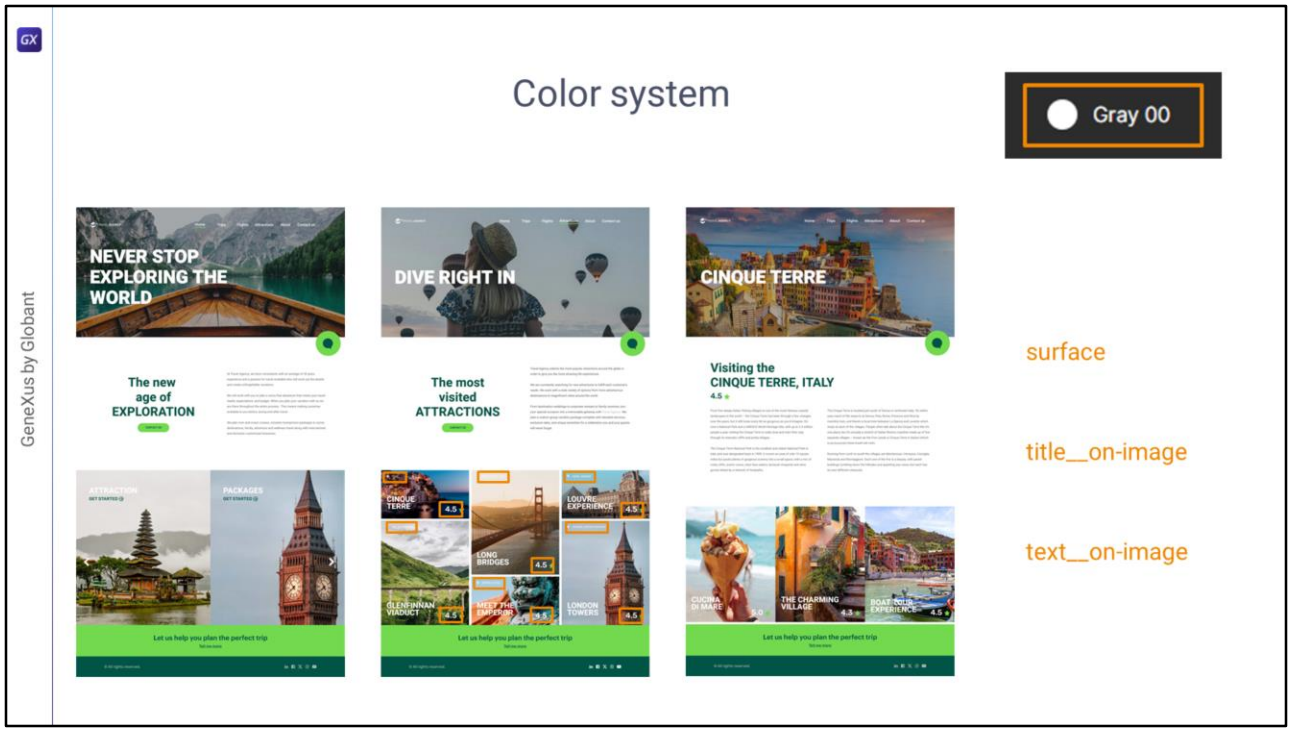
The color system could represent the function of colors in the application in a much more semantic way. For example, this white, Gray00...



...could be used both as a background color and...



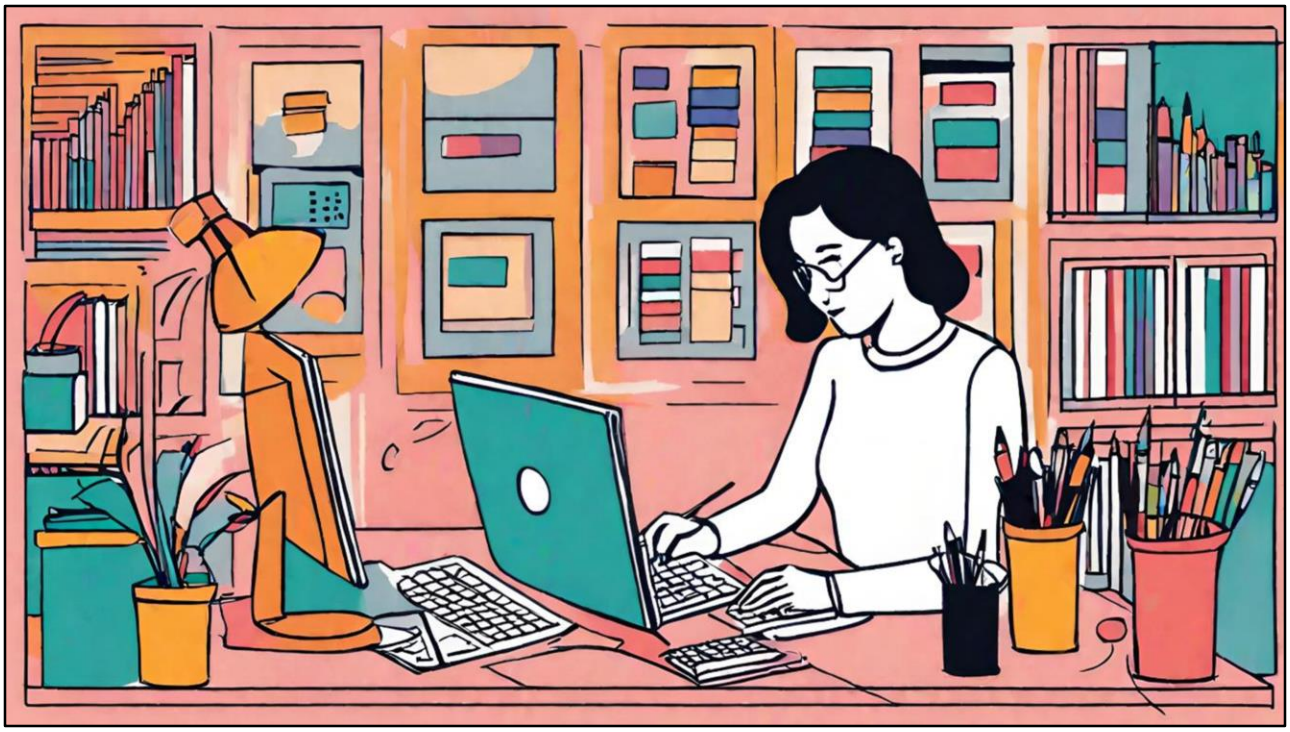
...as the color of a **title on an image**, both of Hero and of Cards...



...as well as for other texts on images, such as those on these Attractions cards. That is, in at least three different functions.

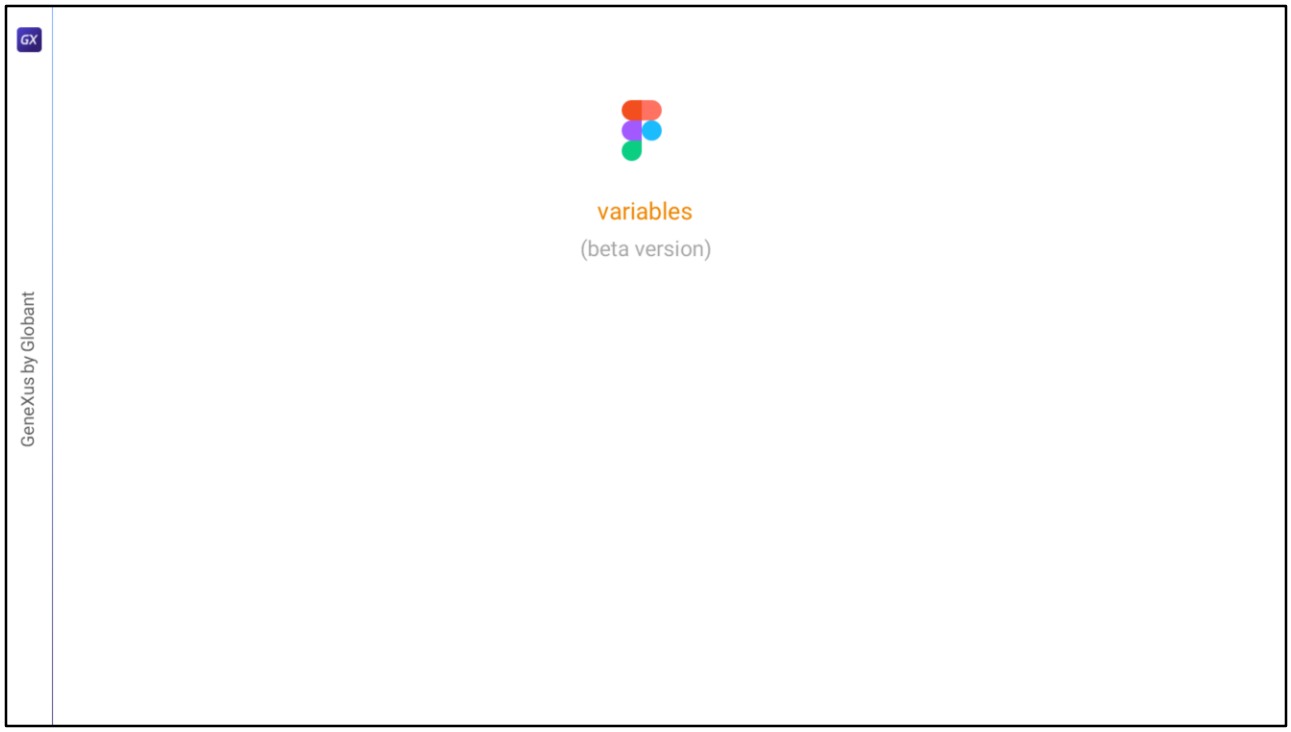
If we give a name to these functions: **surface** color, color of **titles** on images, and color of **texts** on images, we are building a color system that is truly more systemic, and of course, more semantic.

It will tell a lot more and in a better way about the application's design. It will then be very easy to change, for example, the background color of the screens. Or the color of the titles on images, or of the texts on images.

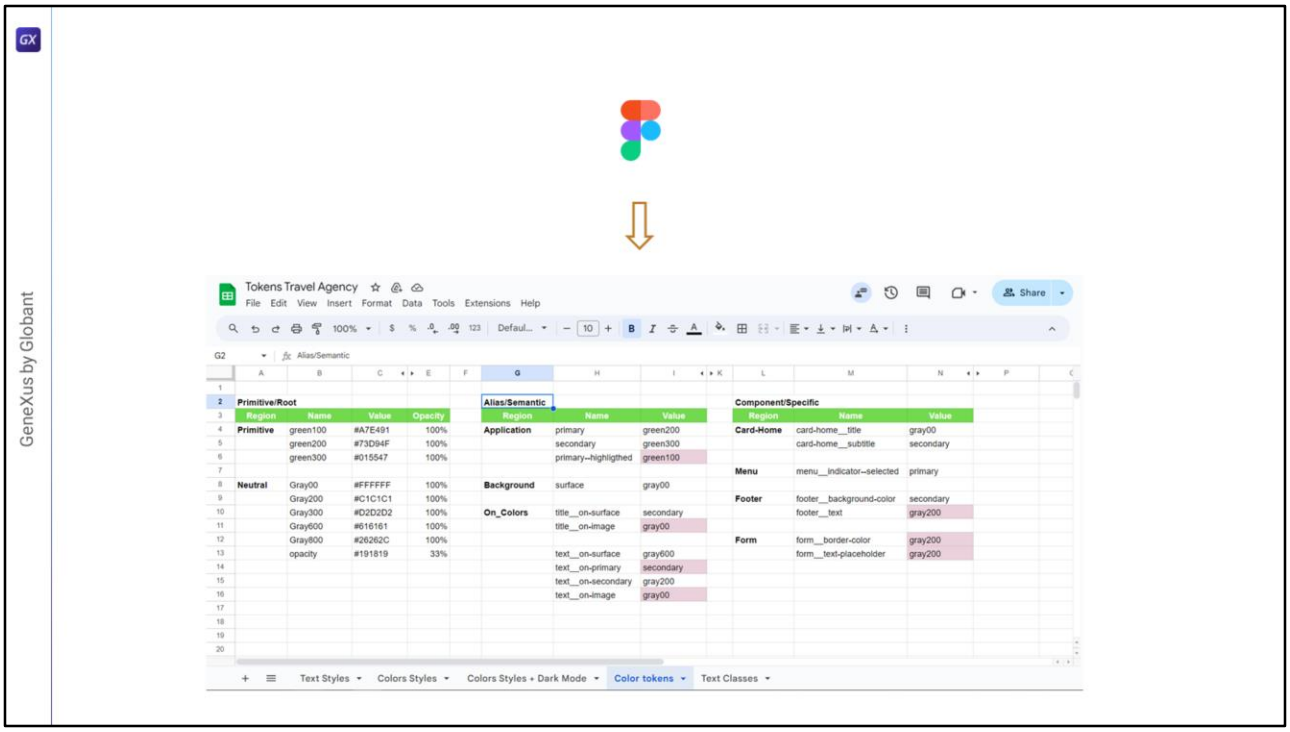


The task of building a good color **system**, that is, of identifying and abstracting its functions, is not so easy, but it is a very important task, which, if it is delayed, will degrade the system in the future. Therefore, it is advisable to focus on solving it as soon as possible, and then everything will go smoothly. At most, minor adjustments will have to be made.

In general, this task is left to the designer, who already works more or less consciously with these abstractions.



If your design tool does not provide that level of expressiveness (this is the case of Figma, which has that option in beta)....



...you can send the frontend developer the system modeling in a spreadsheet, for example.

This doesn't mean that we, frontend developers, even if we are not the ones building the model, have to understand it. Of course, if we don't have a designer, or the designer can't handle the systemization, we will have to do it ourselves.

So let's analyze this model that Chechu put together for me (and that I have been completing) for the color system of our application.

First, we can see that it is designed in 3 levels of abstraction. The most basic, this one, corresponds to the color palette, where we simply give a name to each color of the palette and not much else.

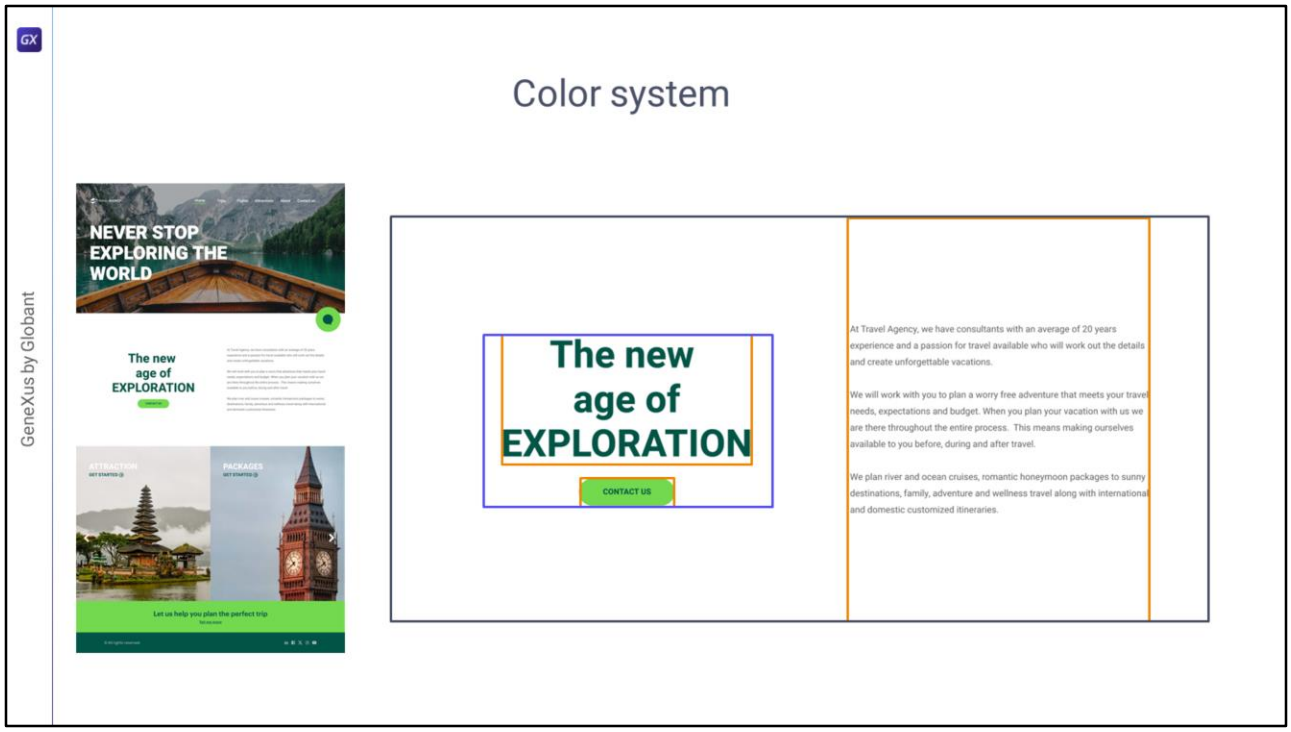
The second level is essential, it is built on the first one, and it is essential because it already corresponds to the global semantics of the colors in the application. It is going to model, then, the color system in the most general way possible.

With the color styles that Chechu had created in Figma so far, we have a very basic combination of these two levels.

Later there may or may not be a third level to specify the special cases that cannot be modeled with the generality given by the second level. That is to say, the second level is thought of as cross-cutting to the entire application, while the third level is much more specific to a particular component or part of the design.

Let's analyze all this a little to understand it. Here we see tokens for the primary color, for the secondary color, for the highlighted primary color (we had already incorporated them before), and we are adding a token for the background color of the screens: this one. Then we see that tokens are being added to represent the colors of the elements that will be placed **on** background colors.

Let's see examples to understand this.

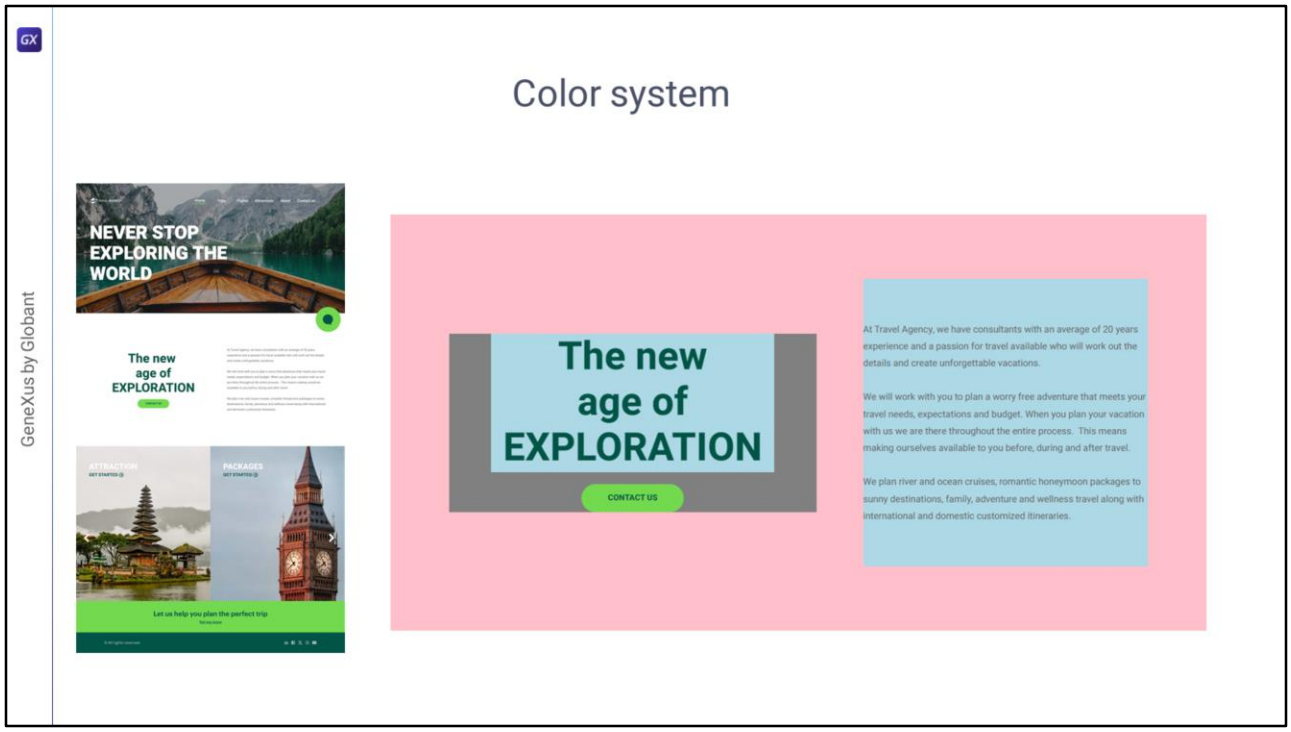


This is because every element of a layout has a background color, the background-color, which may be transparent.

For example, these two texts have a transparent background color, but this button does not.

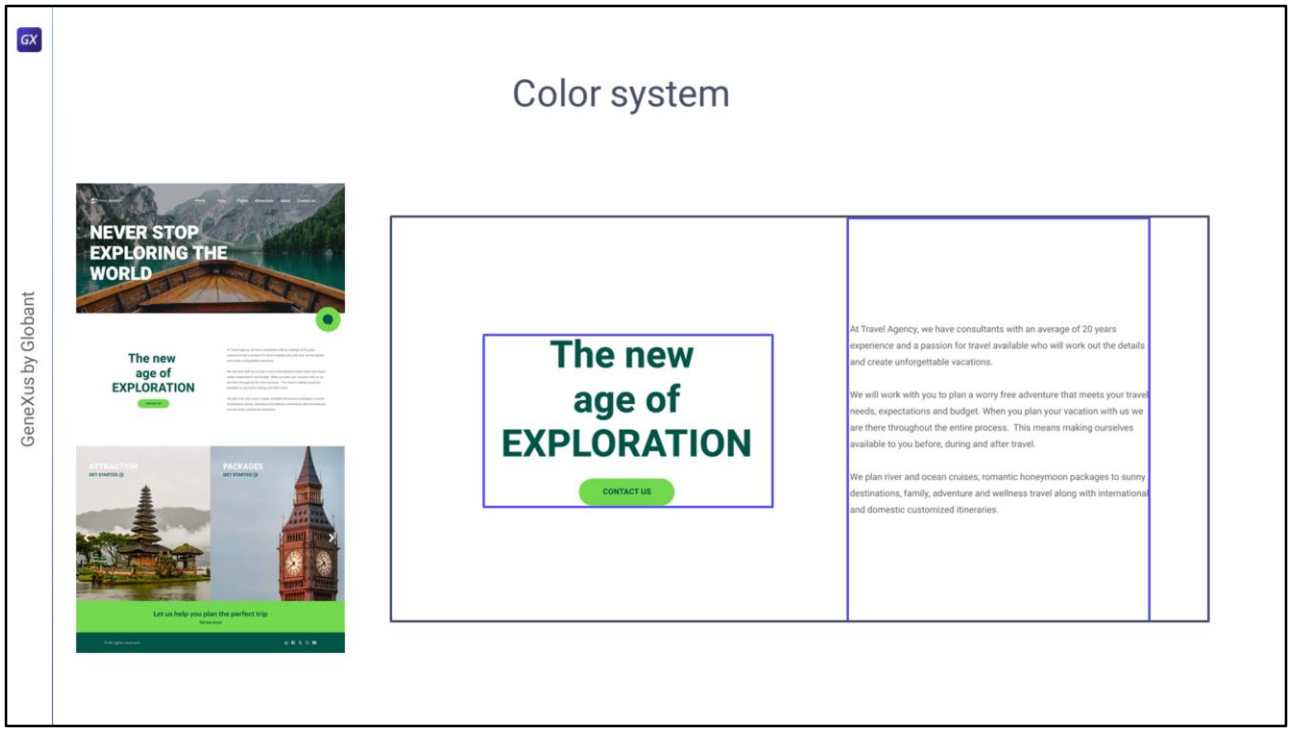
But these two controls are in turn inside another one – table or flex, it doesn't matter – which can also have a background color. Or, be transparent.

And this one and this one, in turn, are also inside a table or flex that may or may not have a background color.



For example, look what happens if I provide different background colors to several of these elements.

In short, in every layout there is a hierarchy of controls, some inside others, each one with a background color, which may (or may not) be transparent.

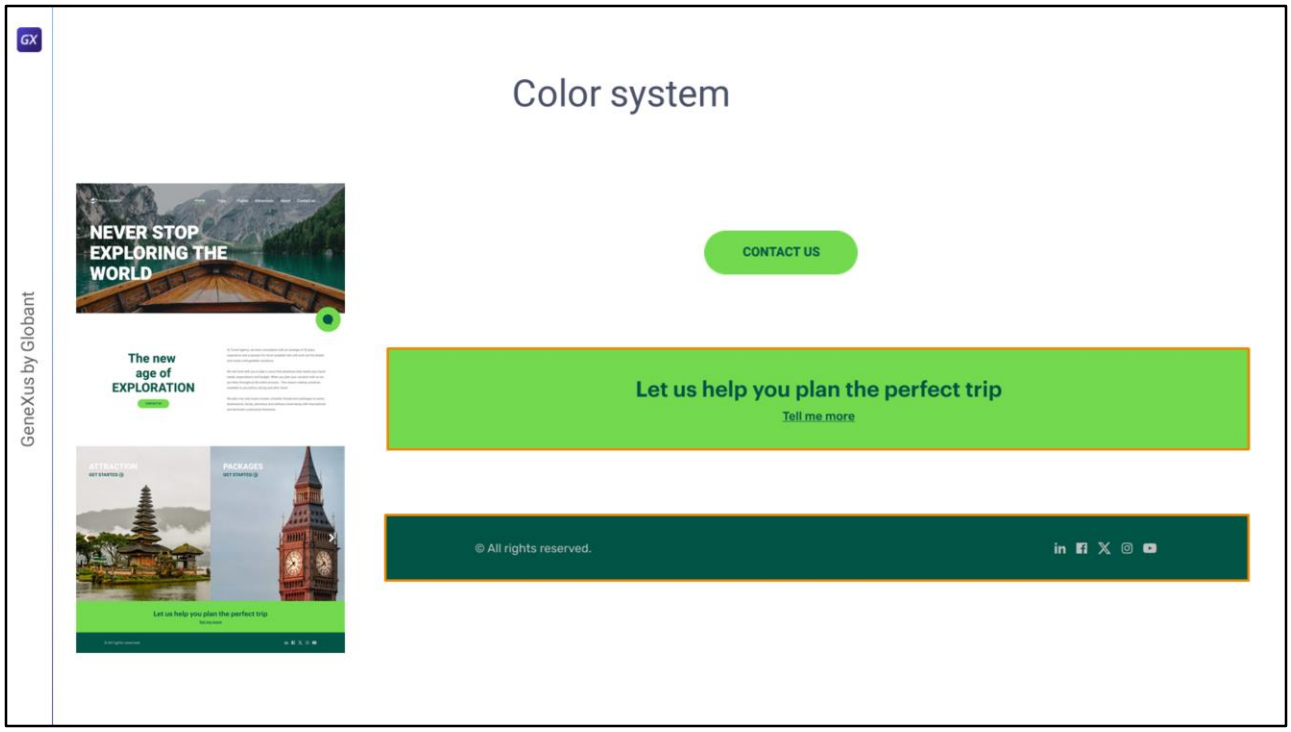


In the example of our design, the only background-color that is not transparent (actually, it is white) is that of the container. Well, except for the button, which has its light green background-color.

What I said a moment ago is not necessary either: the background-color of this table could also be transparent, if the one that contains it has a white background-color.

It is our case; do you remember that we had placed the background-color at the level of the Application class in the DSO, so that it was universally valid for all layouts? Well, if you don't remember, don't worry, that is not the point now.

What I want to pay attention to is the contrast required between the background and the foreground.

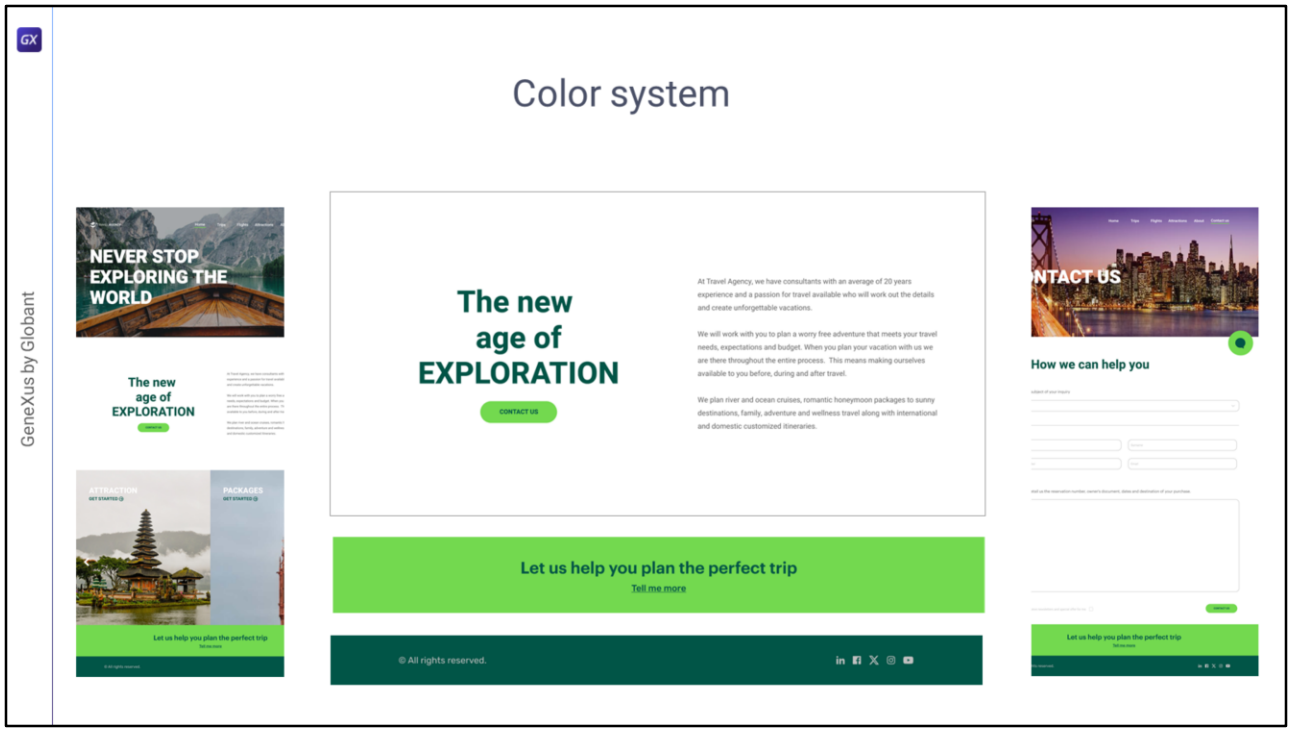


This one, for example, can be a table or a cell of a table, and this another one. And that table or cell has a background-color. In one case the light green, in the other the darker green.

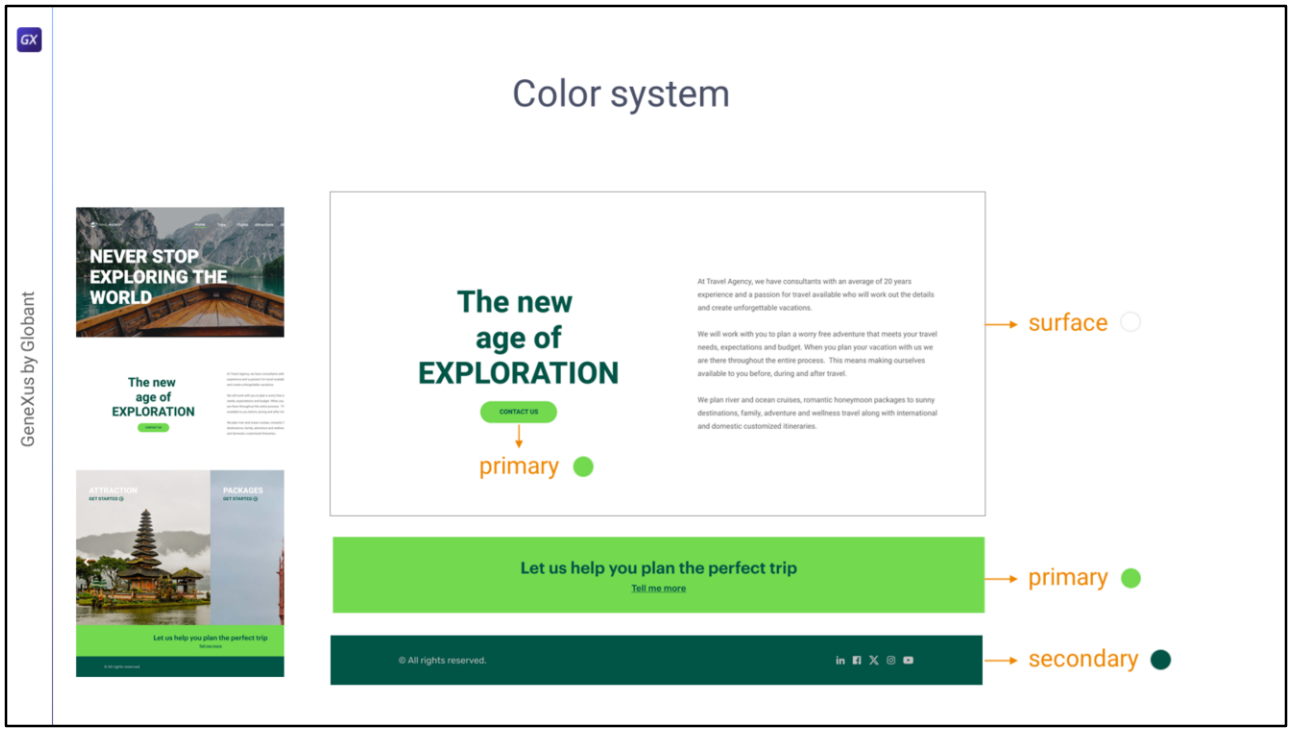
The controls that are inside have a transparent background-color to be placed **on** that surface that is the background.

But they also have their own color; in these cases of texts they are the **color of the text**, which has to contrast against that **background surface**, and that is all the designer is interested in: to achieve a good contrast between the superimposed elements so that they can be displayed properly and are not confusing.

The button can also be thought of in this way, as having a background-color and the text of the button **contrasting on** it.



So if we analyze the system from this perspective, we can see 3 background colors at work.

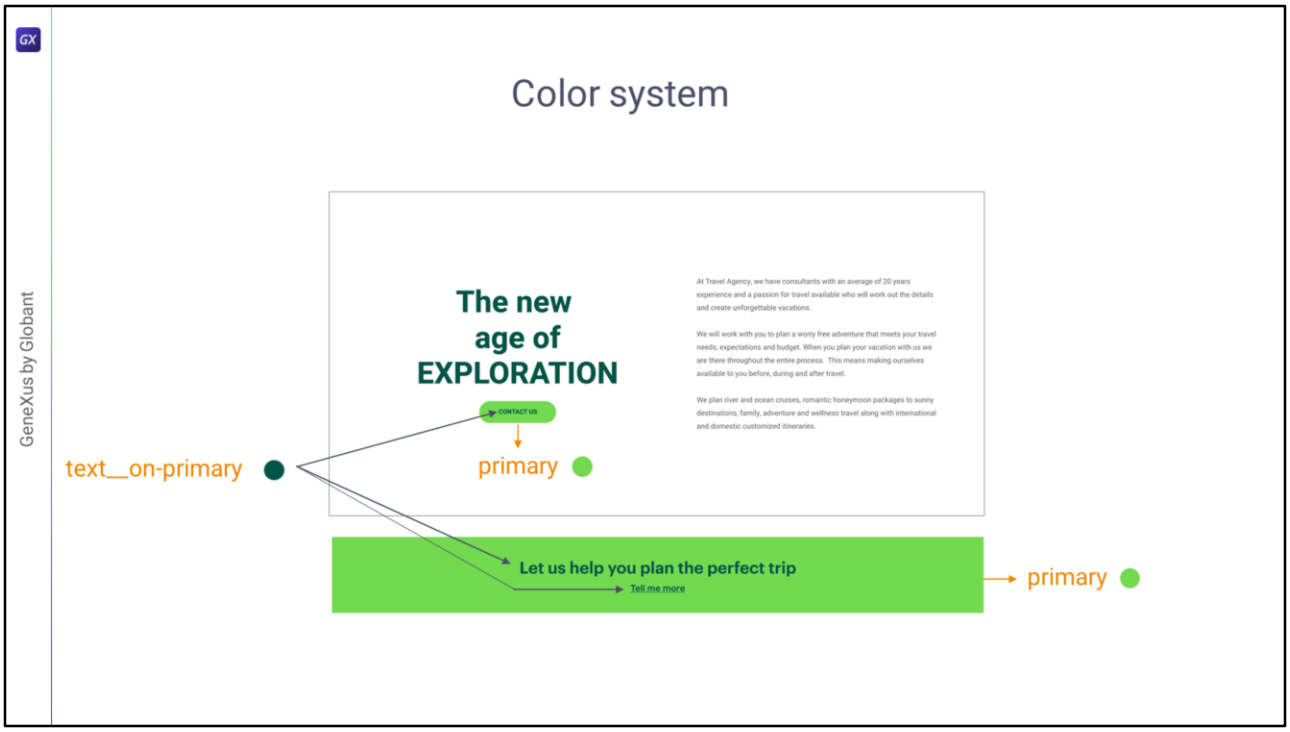


This one, which we will call **surface**, will correspond to the white color of the color palette.

This one, which we call **primary**, is a green color of the color palette (light green).

And this one, which we call **secondary**, corresponds to the other green.

For the button, we also have the **primary** background color.

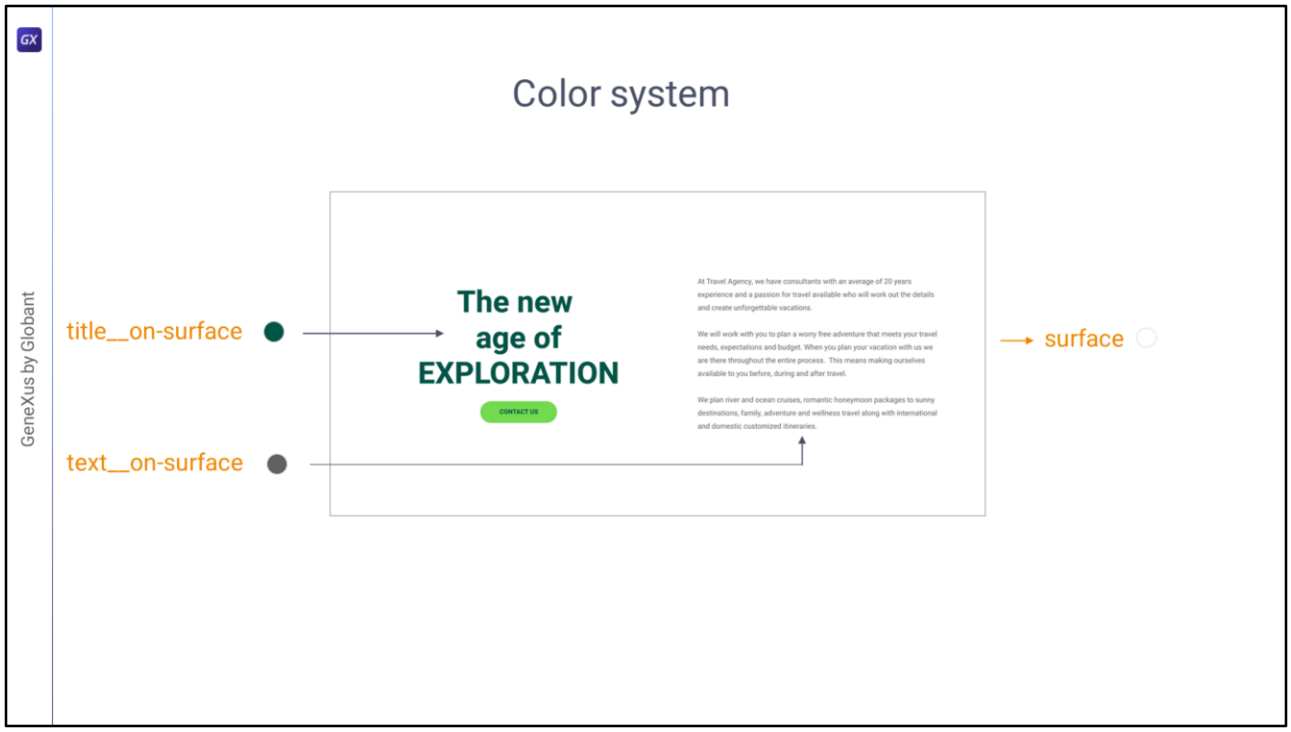


But, as we were saying, a good contrast between the colors placed **on** these **background colors** is essential to achieve a successful design.

If we look at the two elements that have a **primary background** (these two) we see that the texts that are placed **on** that **primary** color correspond to the same color.

Therefore, we can think of a token that specifies the color of the texts that are placed on a **primary** background. And that token is the one we call **text__on-primary**. (Note that I am already using the BEM naming convention to name that token).

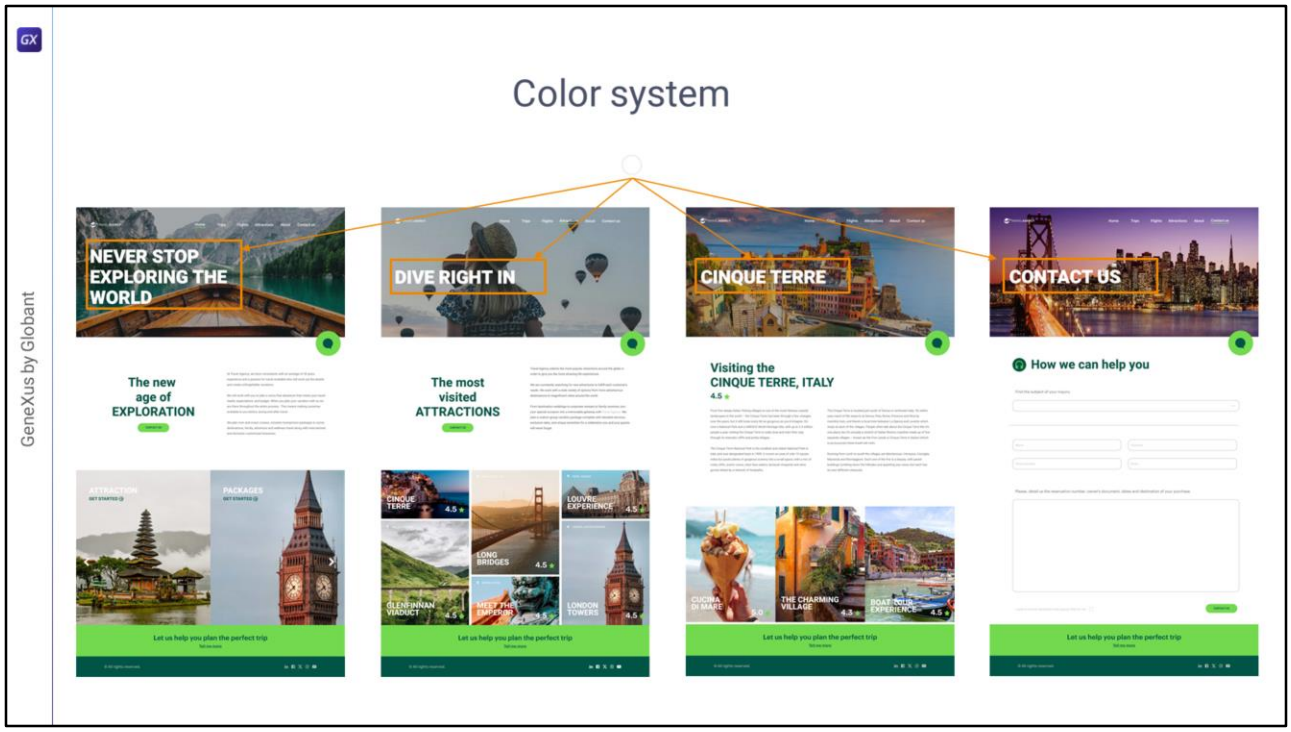
In this case, the value we will give to the token is that of another token, the **secondary** one, but, beyond that, it doesn't matter what value we give it, what matters is that we have just isolated, identified, a token that will perform this function: to be text placed **on** a **primary** surface.



And while we are here, we can also think about the colors of the texts that will be placed **on** the **surface**. We see two types of differentiated elements:

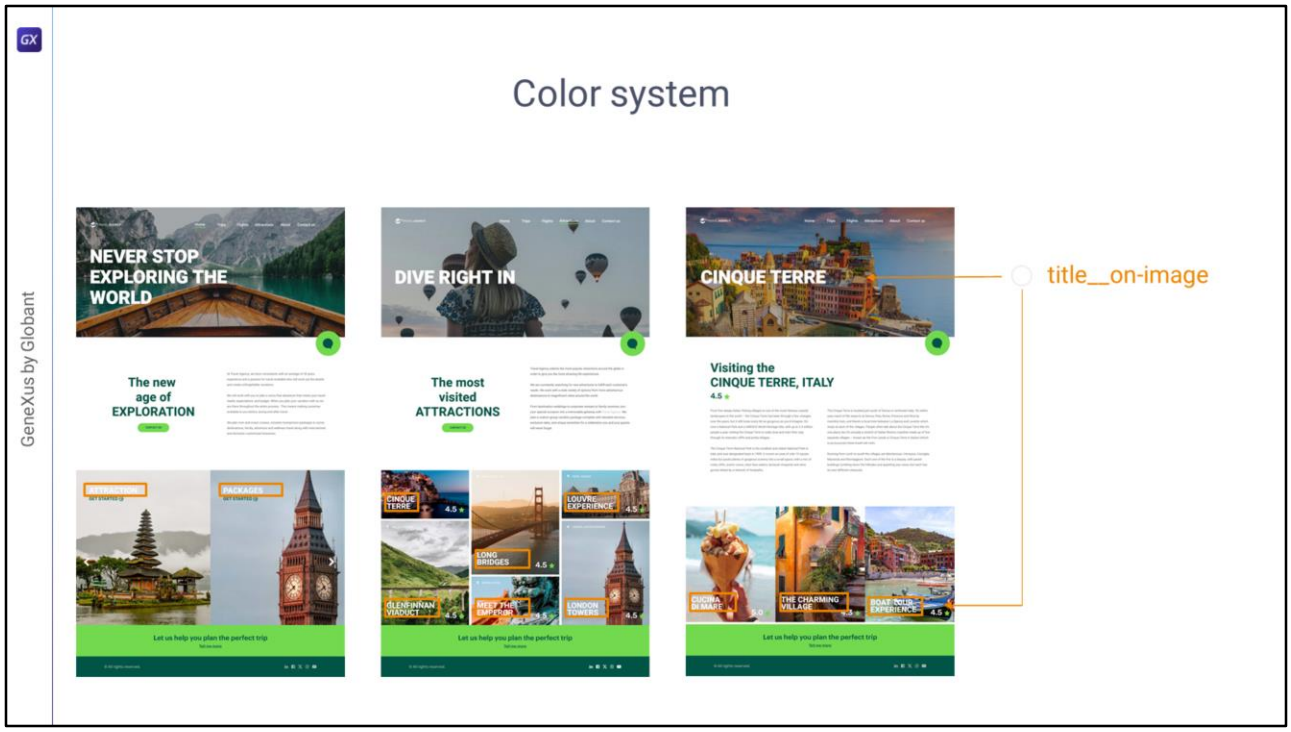
The **color of the title**, and the **color of a** common **text**. Here I'm not sure if it is clear that they are two different colors. But even if they were not, it is convenient to differentiate them because their functions are different; even if for the current design they were combined, which is not the case, because they are two different colors. But even if they were combined with the same value, it would still be good to separate them into two elements because they could change later and one could take a value and the other another value, as they correspond to different concepts.

So we can give them names that stand for the two different concepts: title__on-surface, a **title** on the surface and a **text** on the surface.

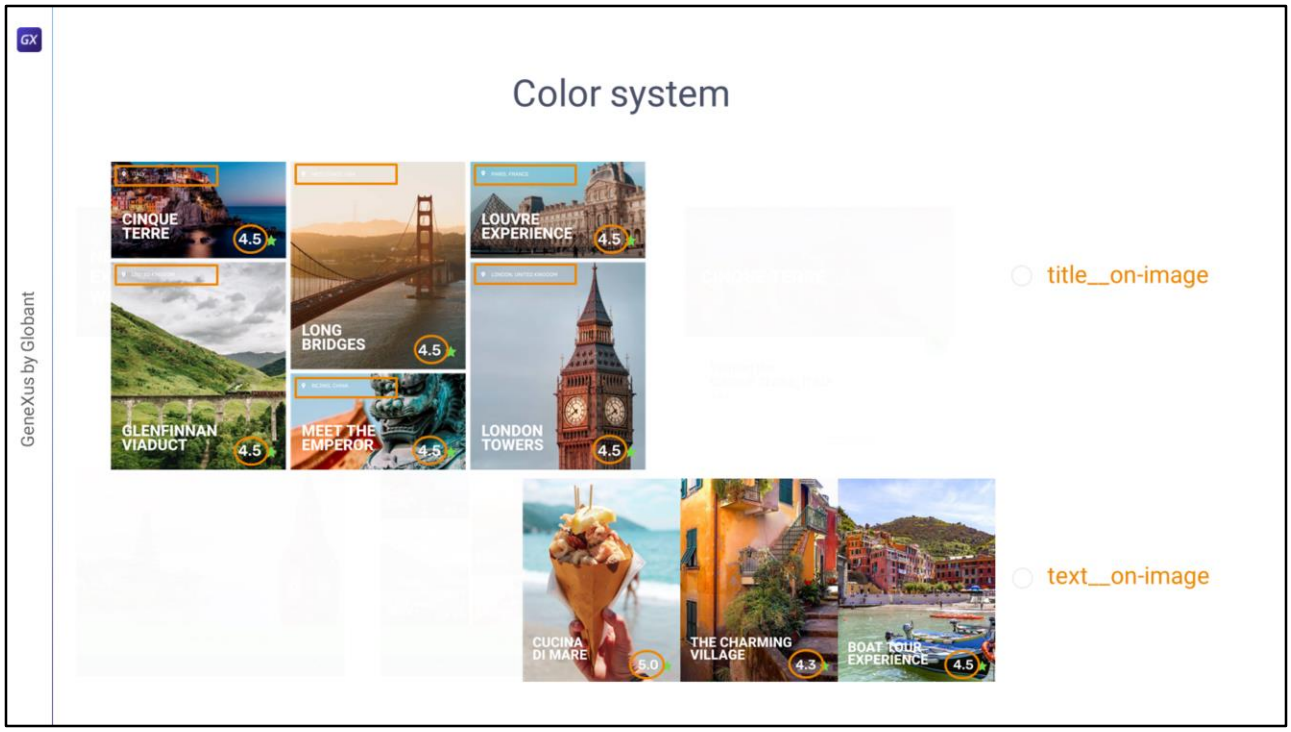


In addition, we can think that **images** also act as **backgrounds** when elements are superimposed on them.
Both the Hero images in the page headers and the ones in all the carousels have texts superimposed on them.

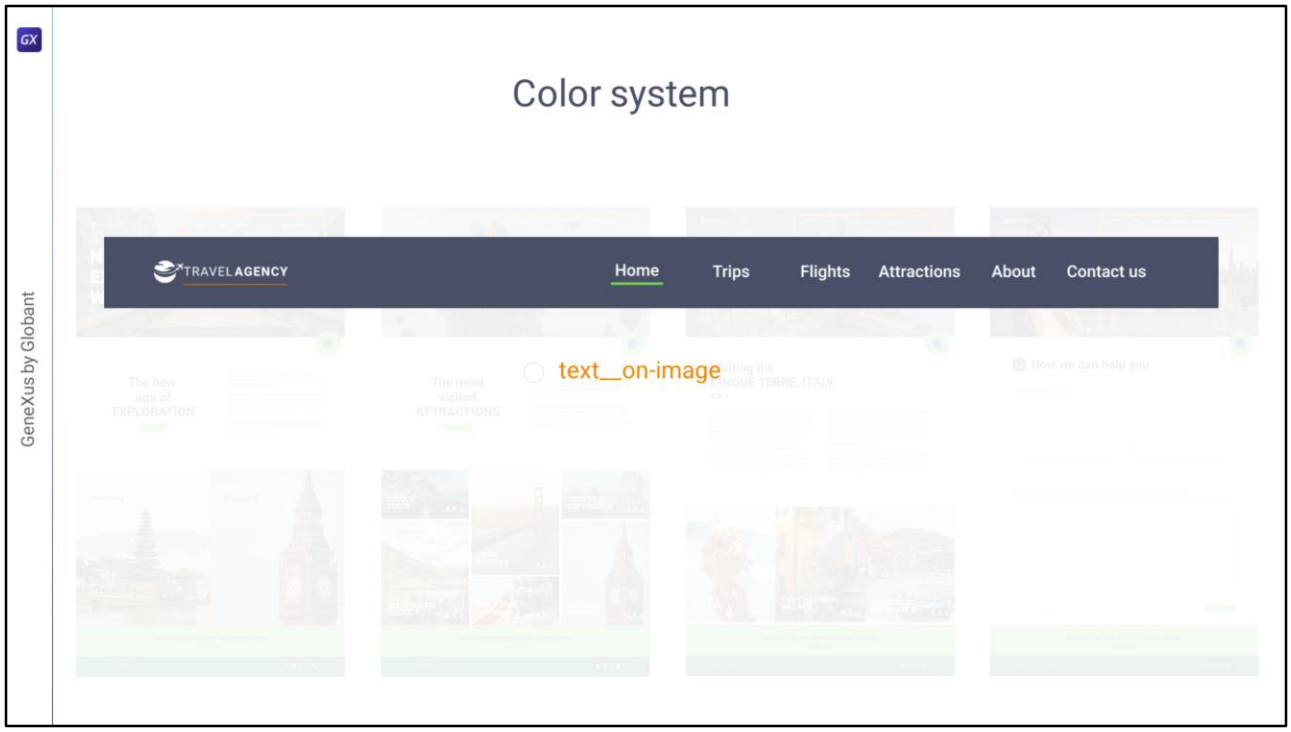
Of what types? We have two cases: that of the **titles** on Hero images...



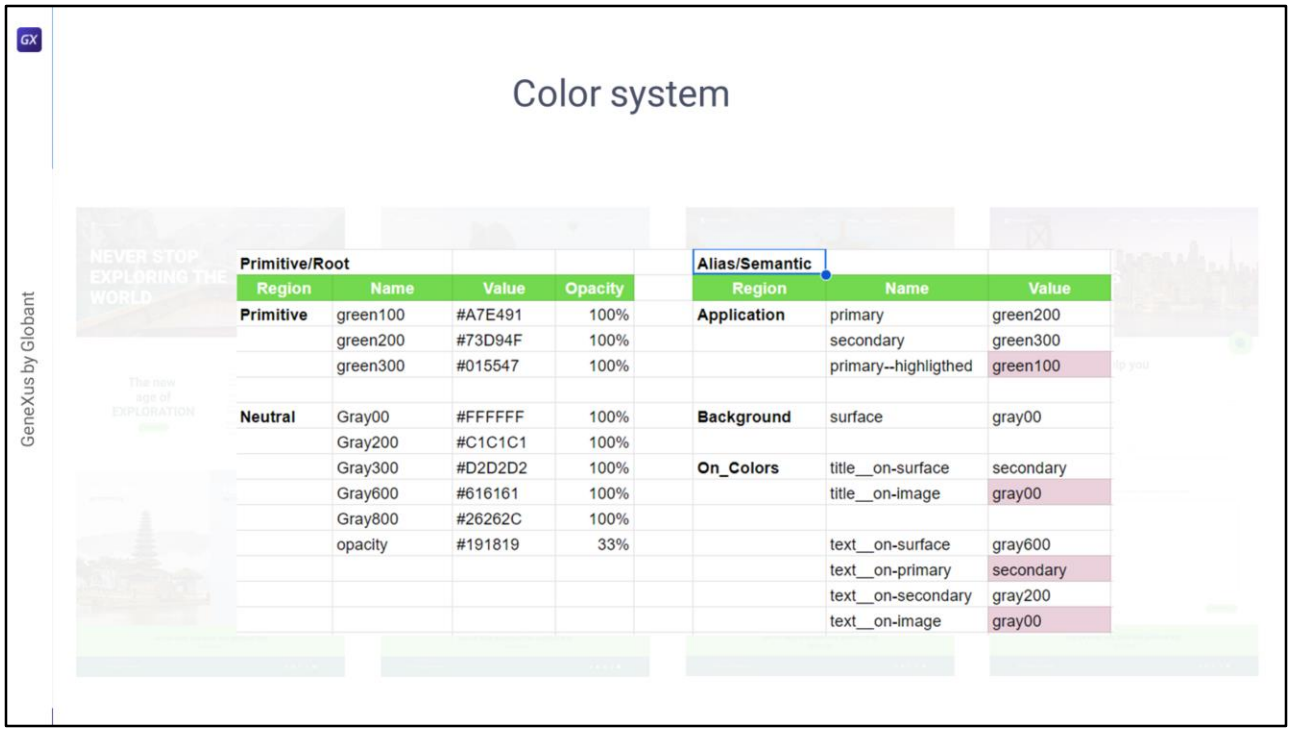
...and on the cards, on one hand. It is the same color in both cases, and surely the designer has thought of it as a "**title** on image" concept (so we can call this color concept **this way** and not separate it into two, at least initially).



On the other hand, the second case that we can identify is the “**text** on image” concept, for all these other cases: those of the rating on these cards and these other ones, and also for the location of the tourist attraction.

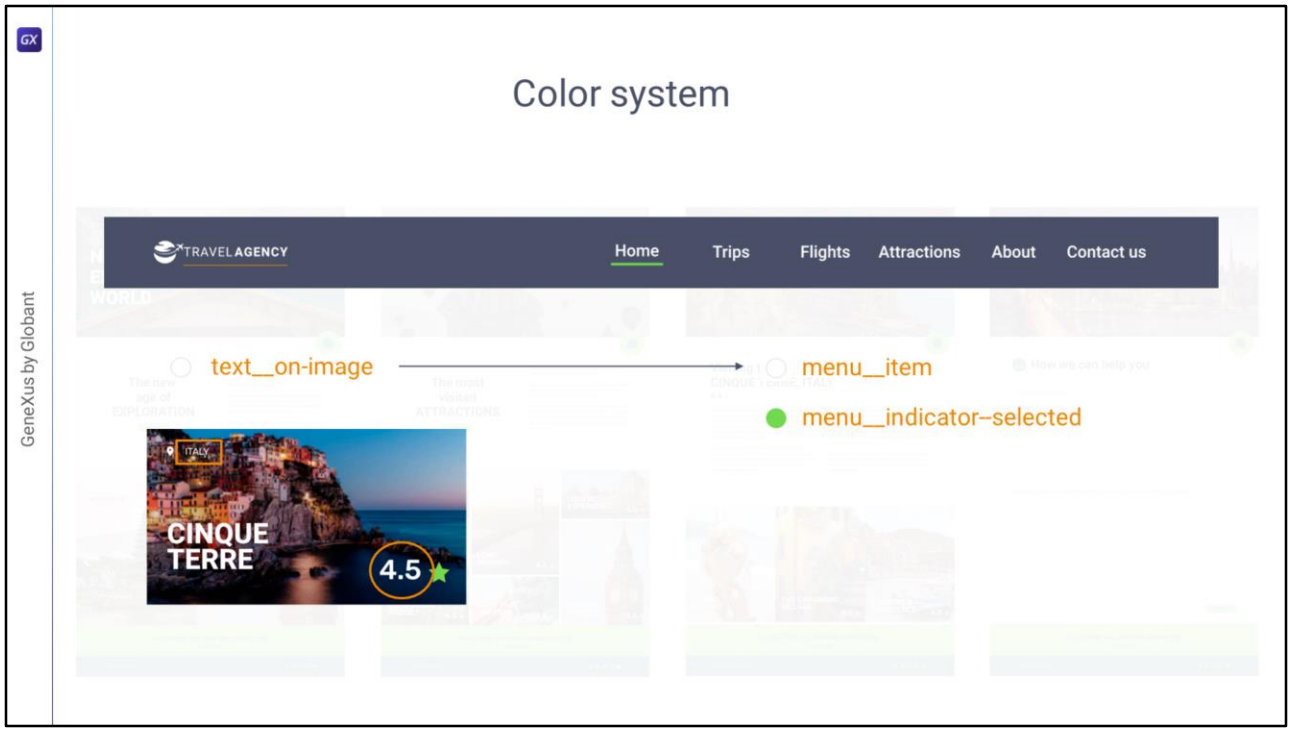


And what about the menu items and these two words in the logo? In principle we could use the same concept.



Having analyzed all this, we already have the minimum semantic level.

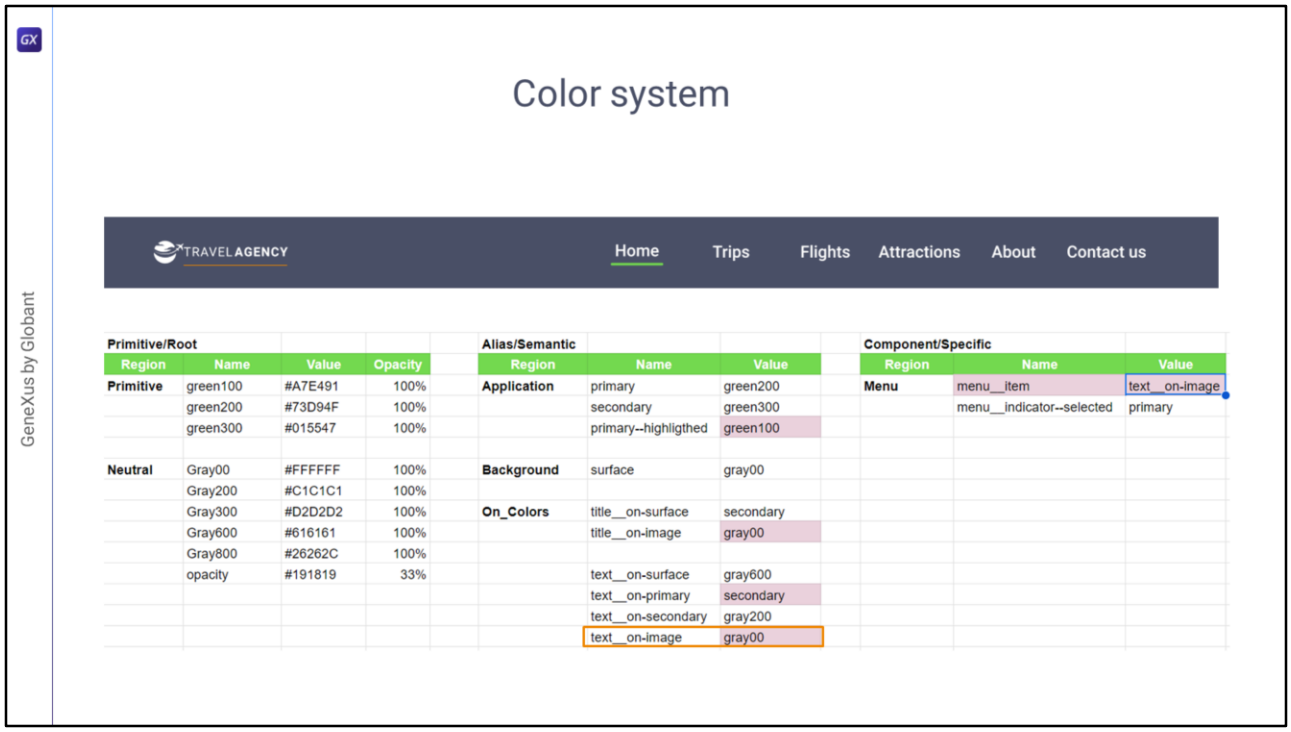
Will we need to add a third – specific – level? And if so, how specific? Or have we already modeled the whole system with this?



For example, as we said, in principle we could use for the menu items the same color as for any other common text that is placed on an image, like these other ones. That is, text__on-image.

But we could also think that the color of a menu item has such a specificity that may deserve to be assigned a color concept of its own. In this design we are implementing the menu to be always on the Hero image, but this decision could change later on. Perhaps we may want to give another color to the menu, so we will need it to be independent from the other uses of “text on image”.

This will be a specific concept of the menu component. It should even add the color of the bar that will be displayed to indicate the selected option, as we see here, which takes this green color, the primary one.



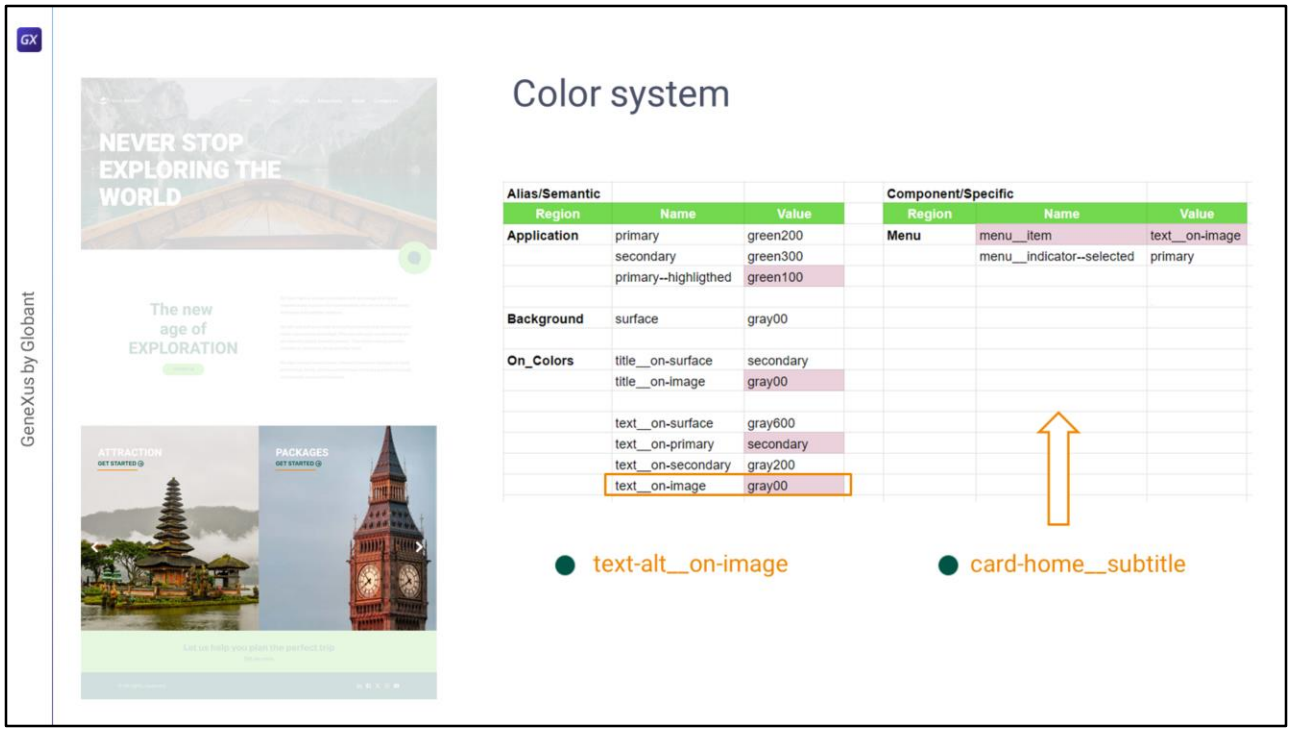
Primitive/Root			
Region	Name	Value	Opacity
Primitive	green100	#A7E491	100%
	green200	#73D94F	100%
	green300	#015547	100%
Neutral	Gray00	#FFFFFF	100%
	Gray200	#C1C1C1	100%
	Gray300	#D2D2D2	100%
	Gray600	#616161	100%
	Gray800	#26262C	100%
	opacity	#191819	33%

Alias/Semantic		
Region	Name	Value
Application	primary	green200
	secondary	green300
	primary--highligthed	green100
Background	surface	gray00
On_Colors	title__on-surface	secondary
	title__on-image	gray00
	text__on-surface	gray600
	text__on-primary	secondary
	text__on-secondary	gray200
	text__on-image	gray00

Component/Specific		
Region	Name	Value
Menu	menu__item	text__on-image
	menu__indicator--selected	primary

Then we would add the third level, because, although these colors will only apply to one component and it wouldn't be so essential to isolate them as tokens, doing so improves the system's maintainability and consistency.
I marked the first one because, actually, I assigned the same value as that of the text__on-image token, which clearly indicates that in principle it would not be necessary that it is independent.

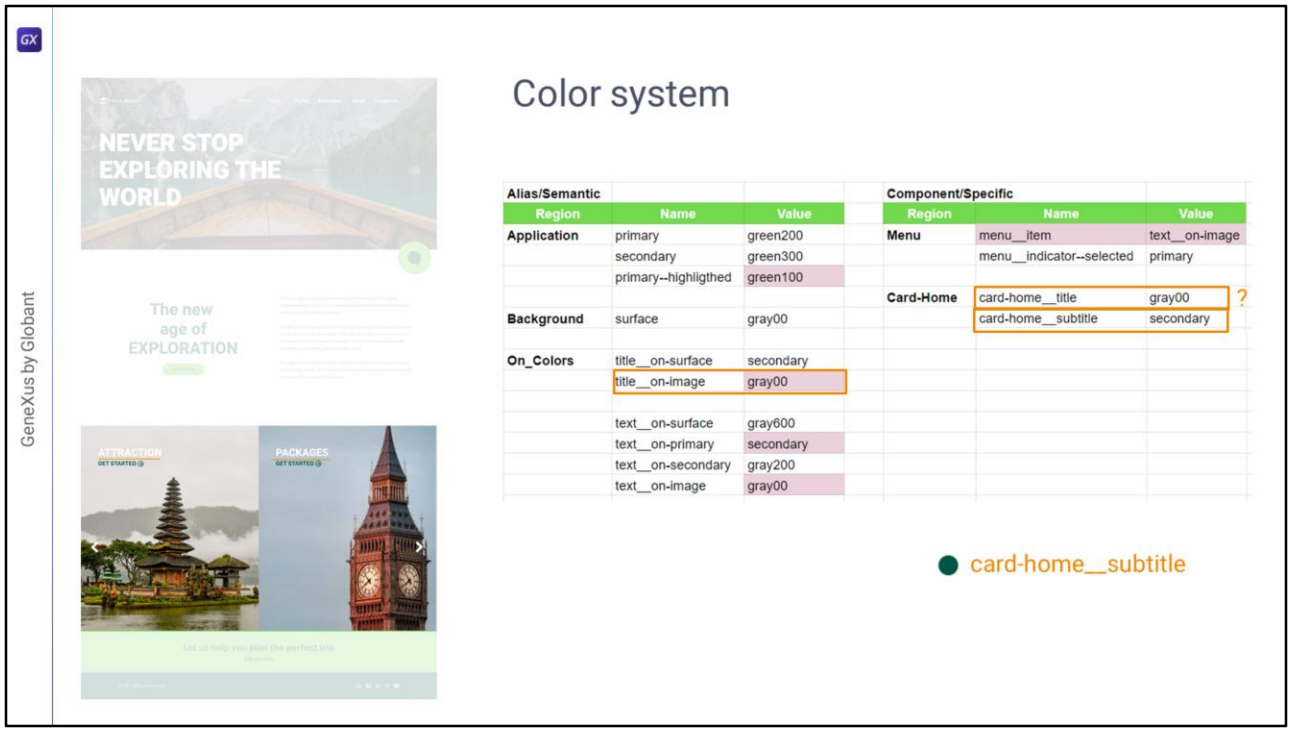
This is the type of decision that we have to make, aiming at a balance. For a small system like ours, in principle there is no disadvantage in this specialization, but as the system grows, the multiplication of particular cases clearly makes it more complex and degrades it. So, I suggest balance!



For example, we haven't modeled the color of this text. It is a text on an image, but not the white that we had for the other cases of text on an image.

So we could define a second, alternative, token for text on an image, at level 2, semantic. Or we could define it specifically as at the level of the component where we will find it, which I will name card-home. Since it acts as a subtitle in that component, I'll call it that.

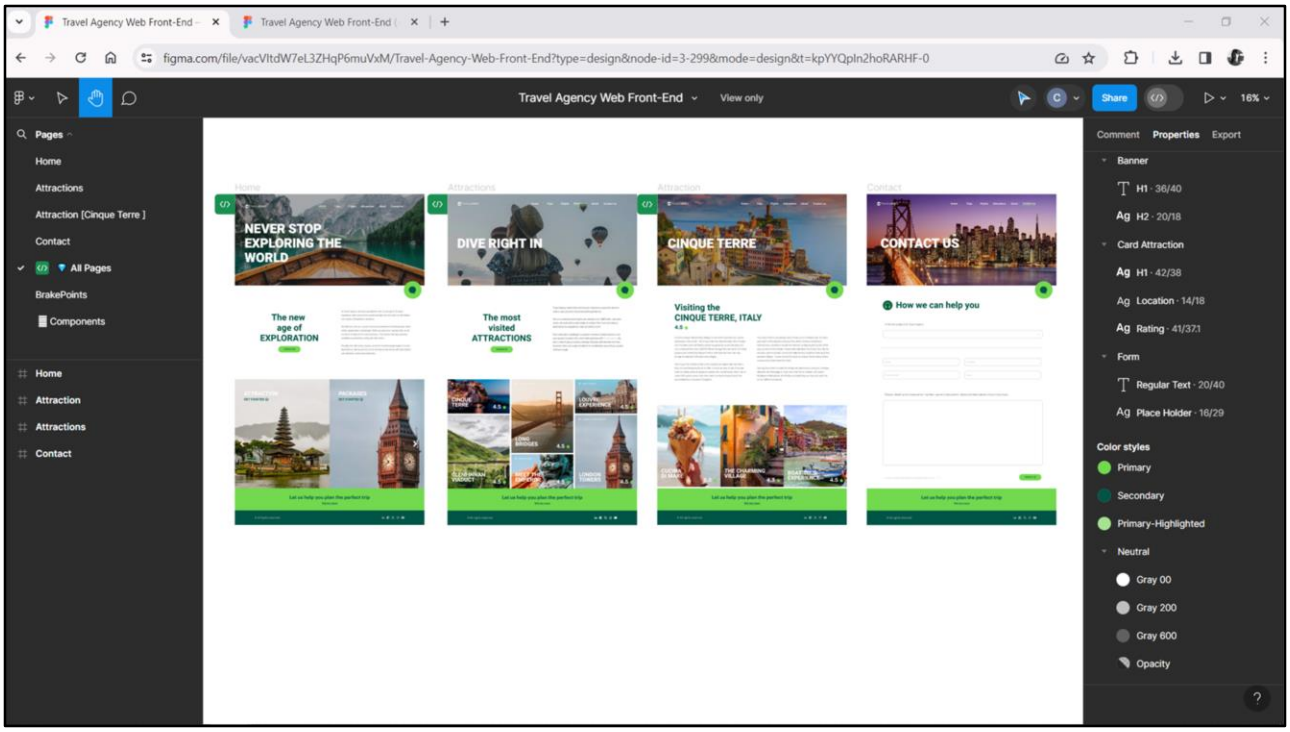
I made this decision because the general token doesn't make much sense since it doesn't seem to be general at all, but to be completely specific to this case.



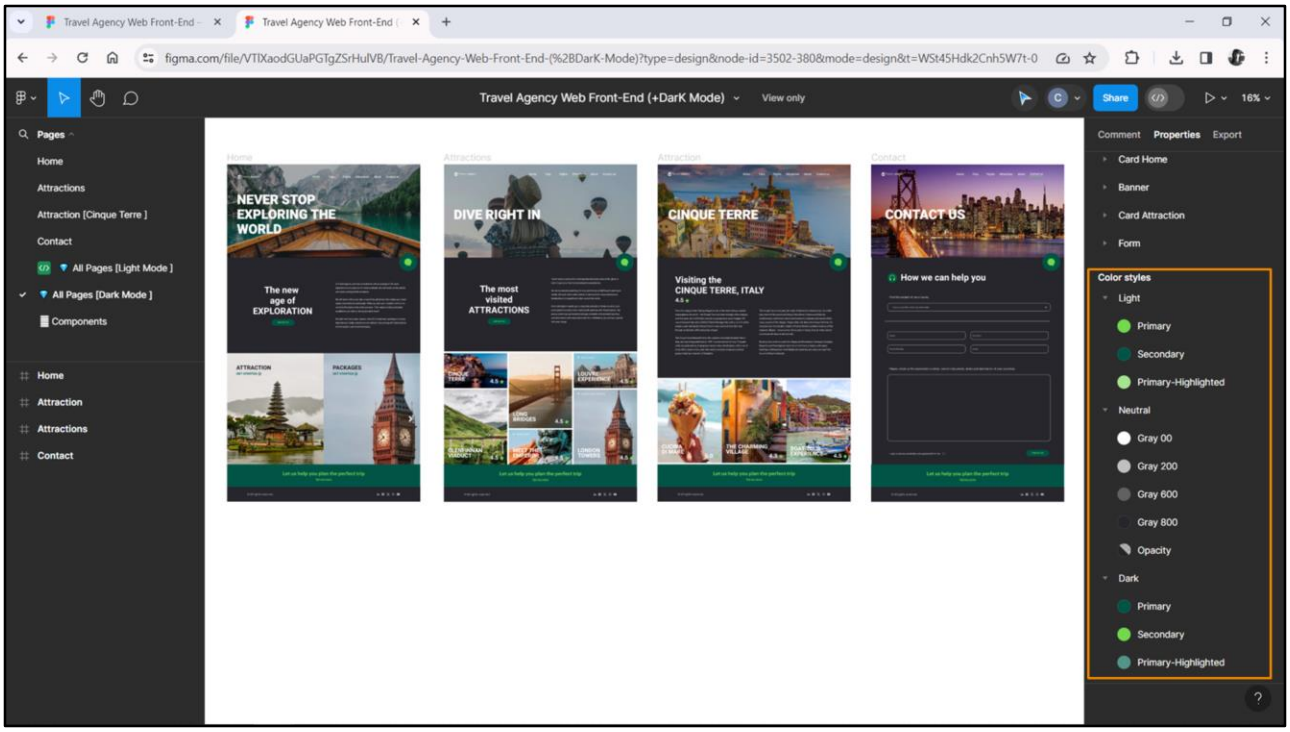
So I added it there, with the value of the secondary token.

You may rightly wonder why for this element I specified a token, card-home__title, if we had already seen that I could use the general one, title__on-image.

It might seem that since I had to define a specific one for the subtitle, then for consistency purposes it would be a good idea to define one for the title. However... it was because of a much stronger reason than that... and it has to do with something that we had not considered so far.



Which is this... This is the Light mode...

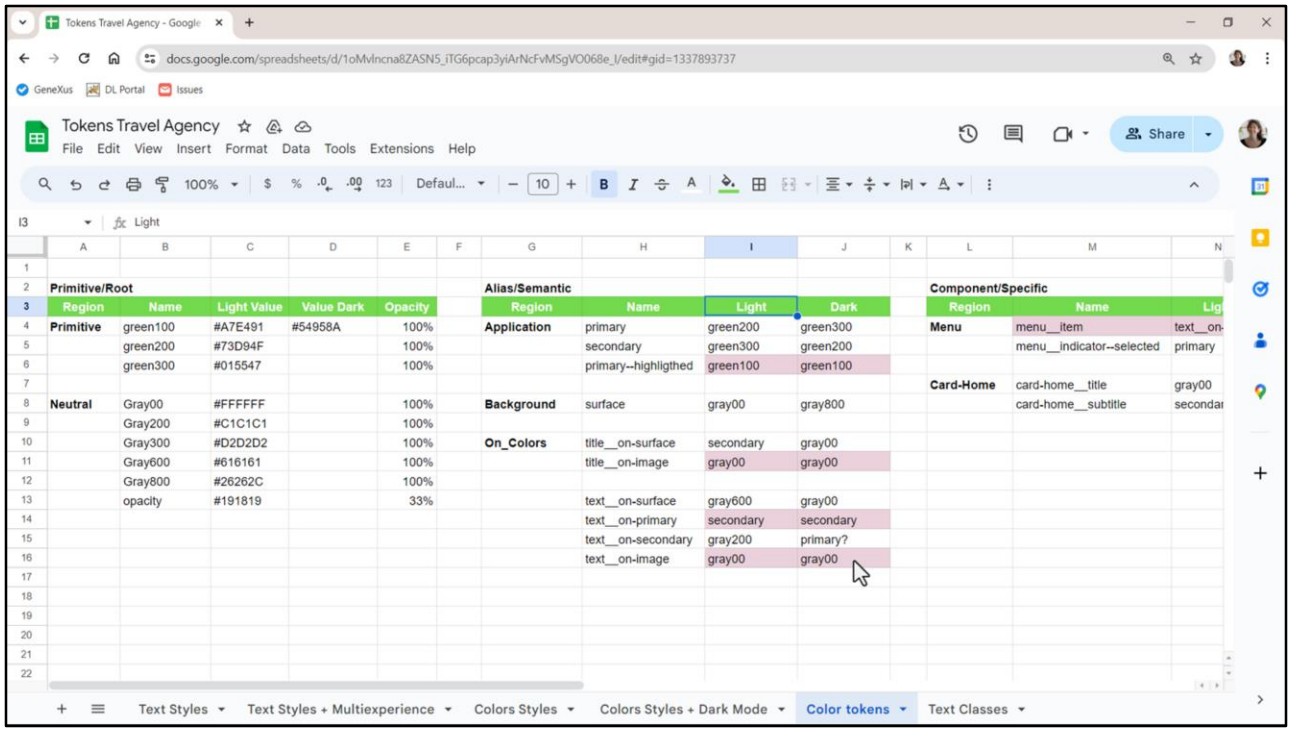


This is the Dark mode.

Chechu made another file, different from the one we were working with, where she added this mode.

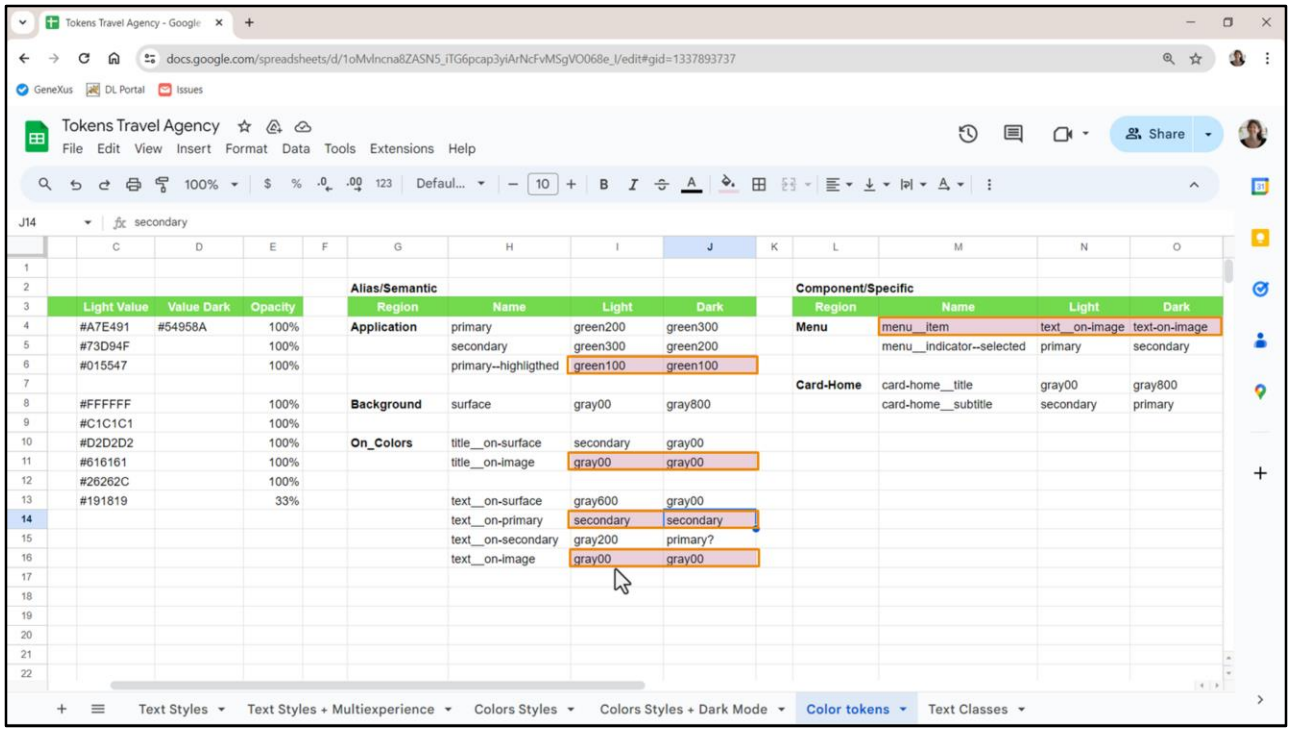
If we click down here and select the properties, on the color styles that we had, now we can see those of the Dark mode.

Although this is Primary Light, in this other one Primary Dark is the secondary one. And the one that is secondary in Dark mode will be the primary one in Light mode.



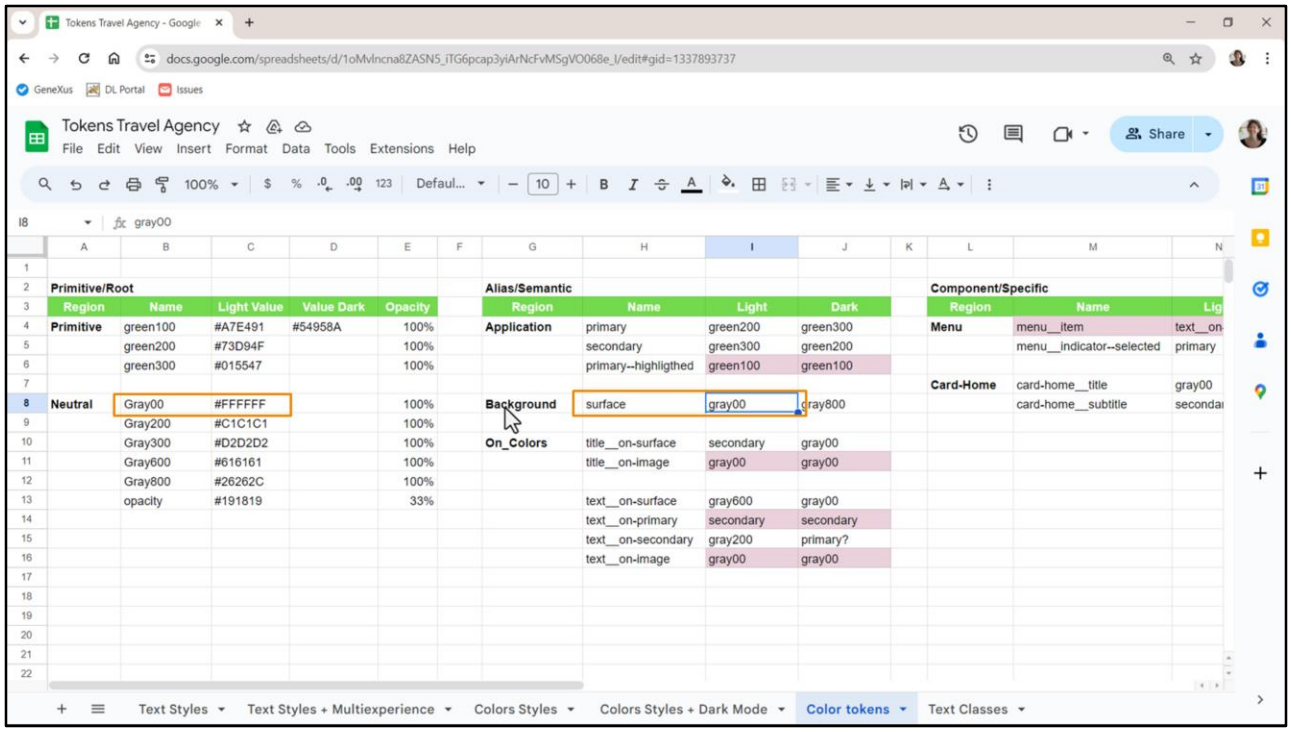
So in the spreadsheet, for each token I will have to specify another column for the value taken by the token for the Dark mode.

The one we had, which said Value (now I changed it to Light) corresponded to Light mode (which was the one we were working on). We have to make these associations: for each token specify what its value is going to be in Dark mode as well.

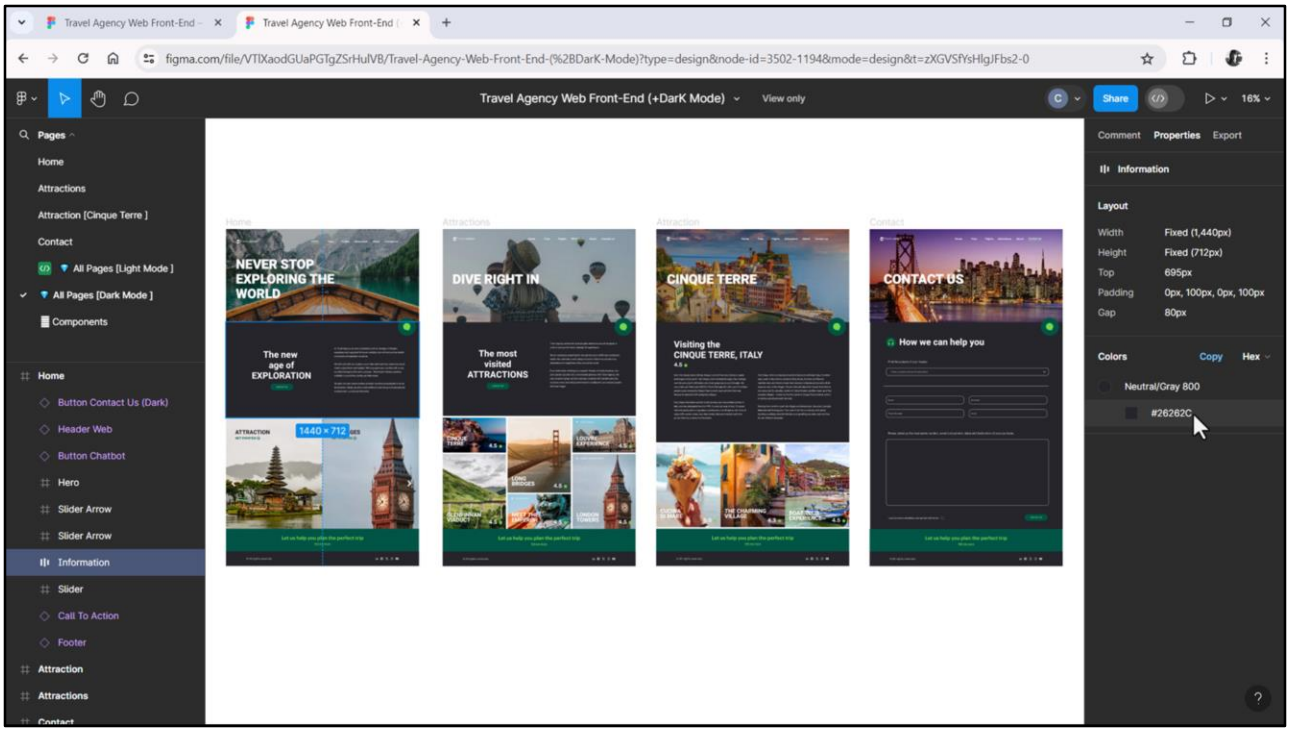


So why did I tell you this? Because if we come here... we would have to inspect all this now to see how the color varies according to the token. What I marked corresponds to when it is the same value in both modes.

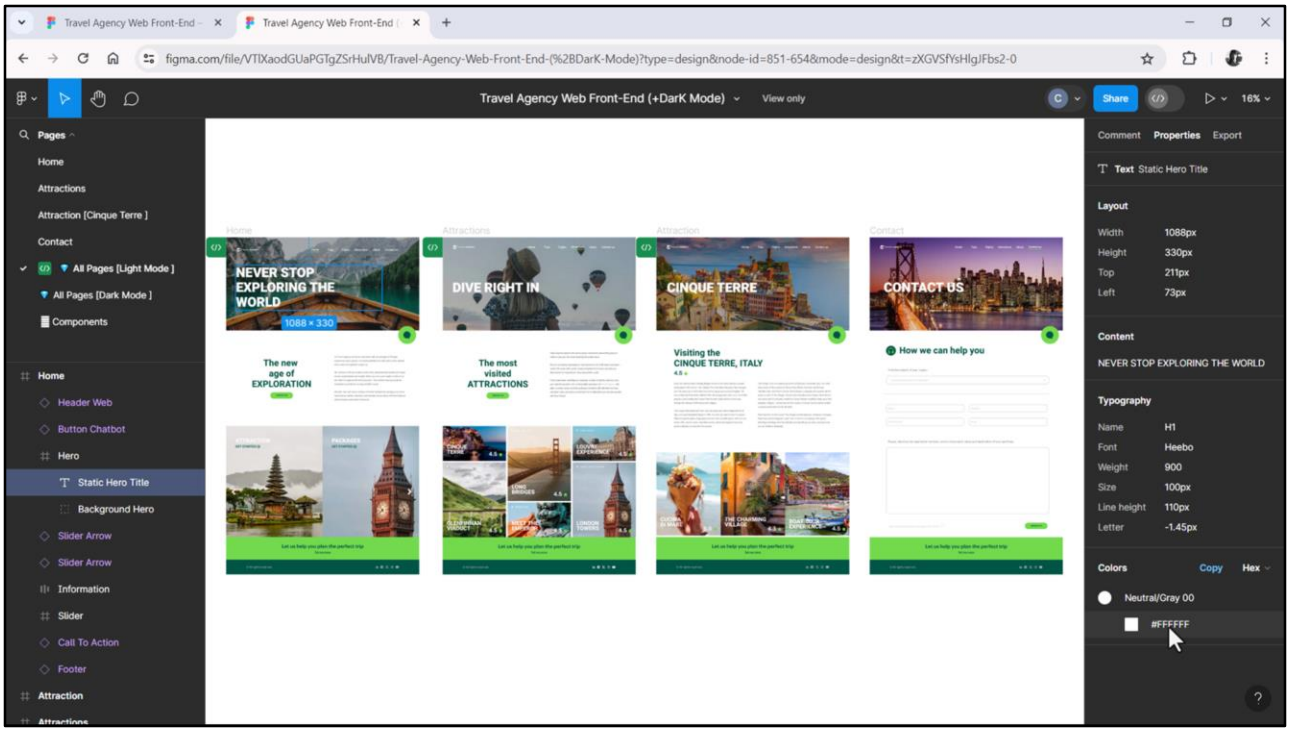
So what is not marked is what varies between one mode and the other.



For example, let's start with the **surface**: while in Light mode it is this white, gray00, which we have here, for the Dark mode it will be this gray800.

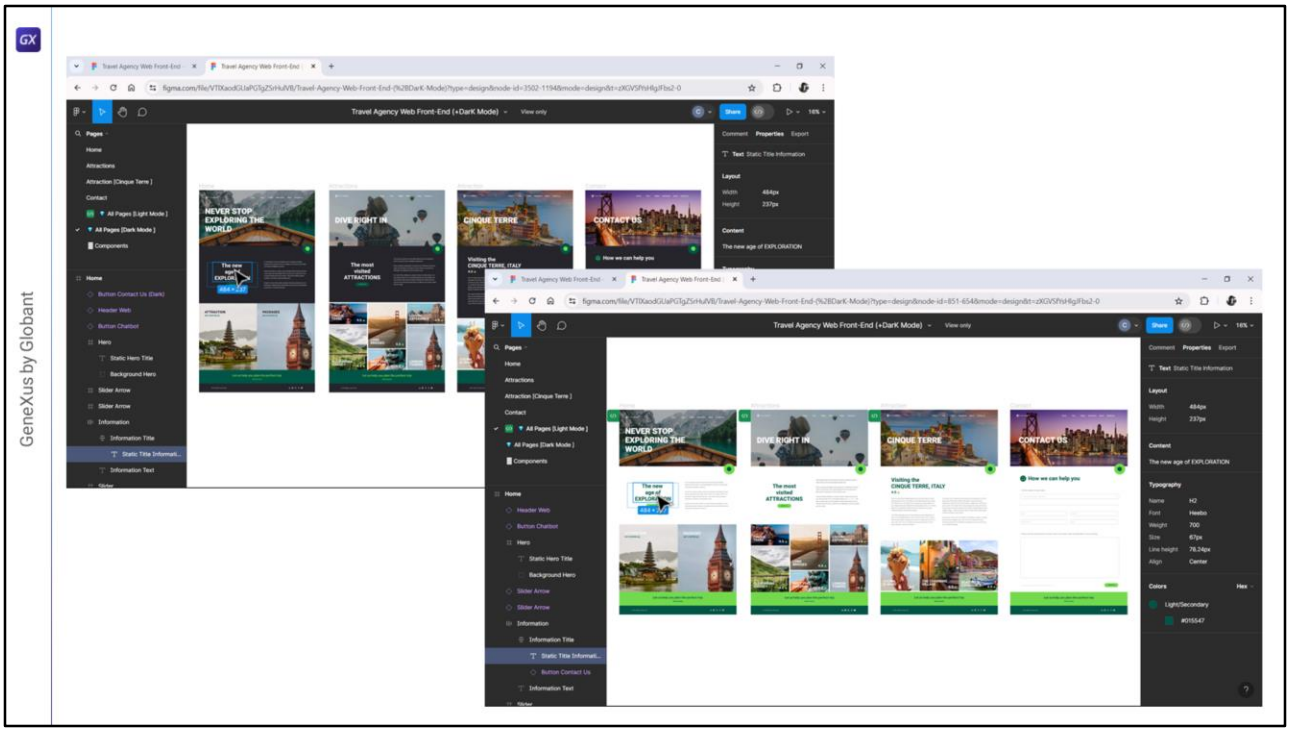


We can see it clearly by selecting this... and we see: gray800... for the Light mode we already knew that it was gray00.

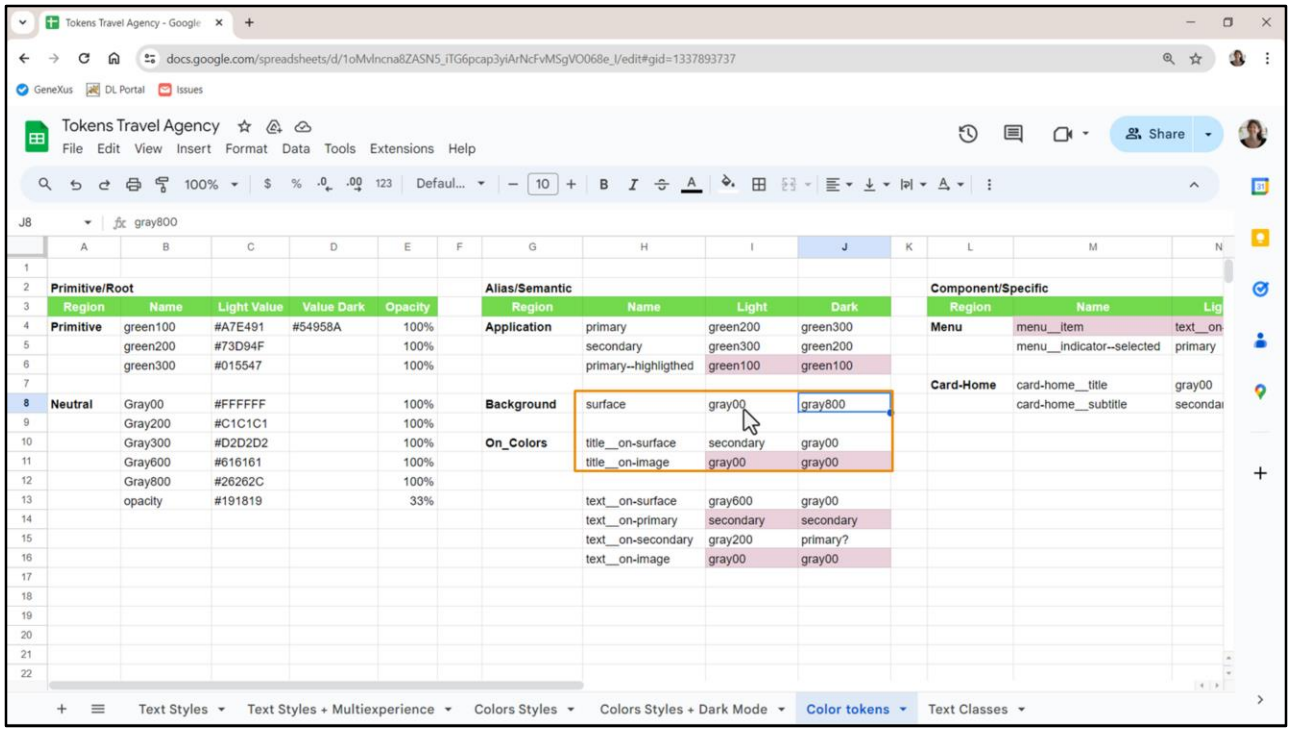


As I was saying, not everything changes.

For example, the color of the title on an image that corresponds to our title__on-image token is gray00 for the Light version ...and for the Dark version as well. It is not modified.

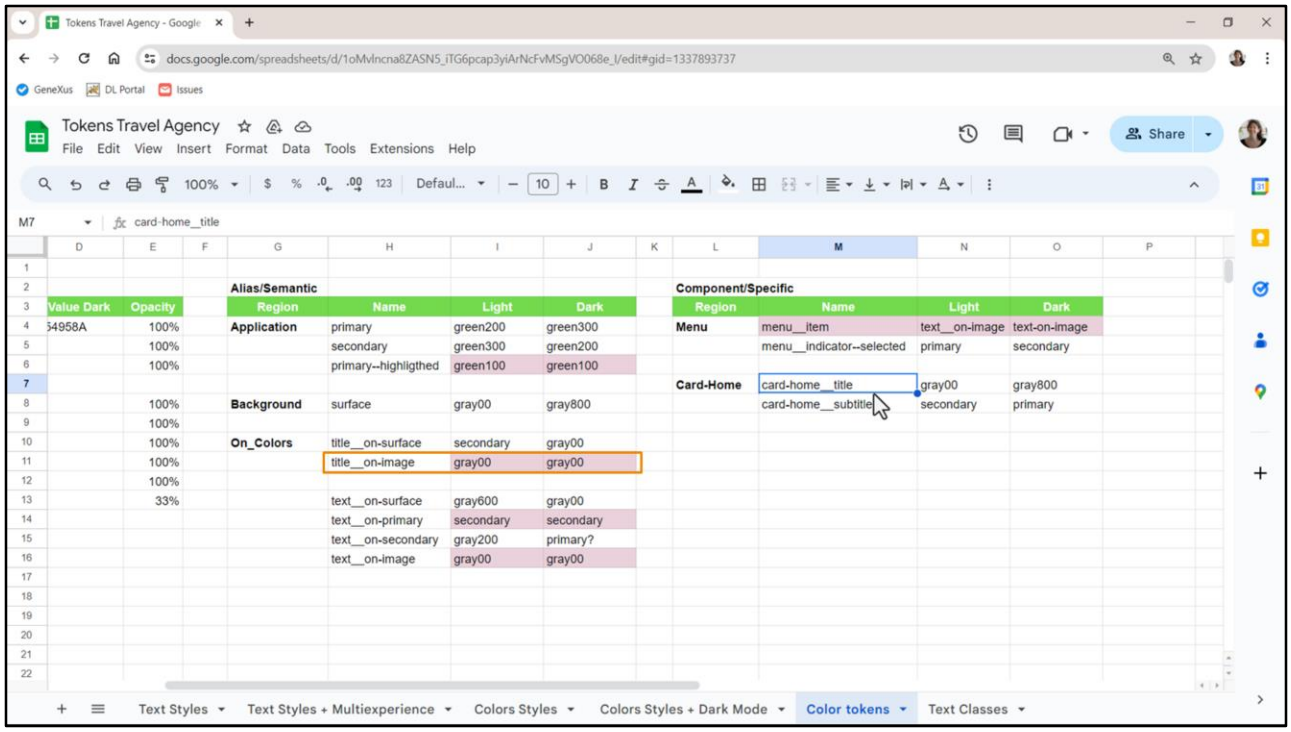


Instead, let's see what happens with the title on a surface, title__on-surface. Here in Dark mode, we can see that it is a white, it is gray00, while in Light mode it was secondary.

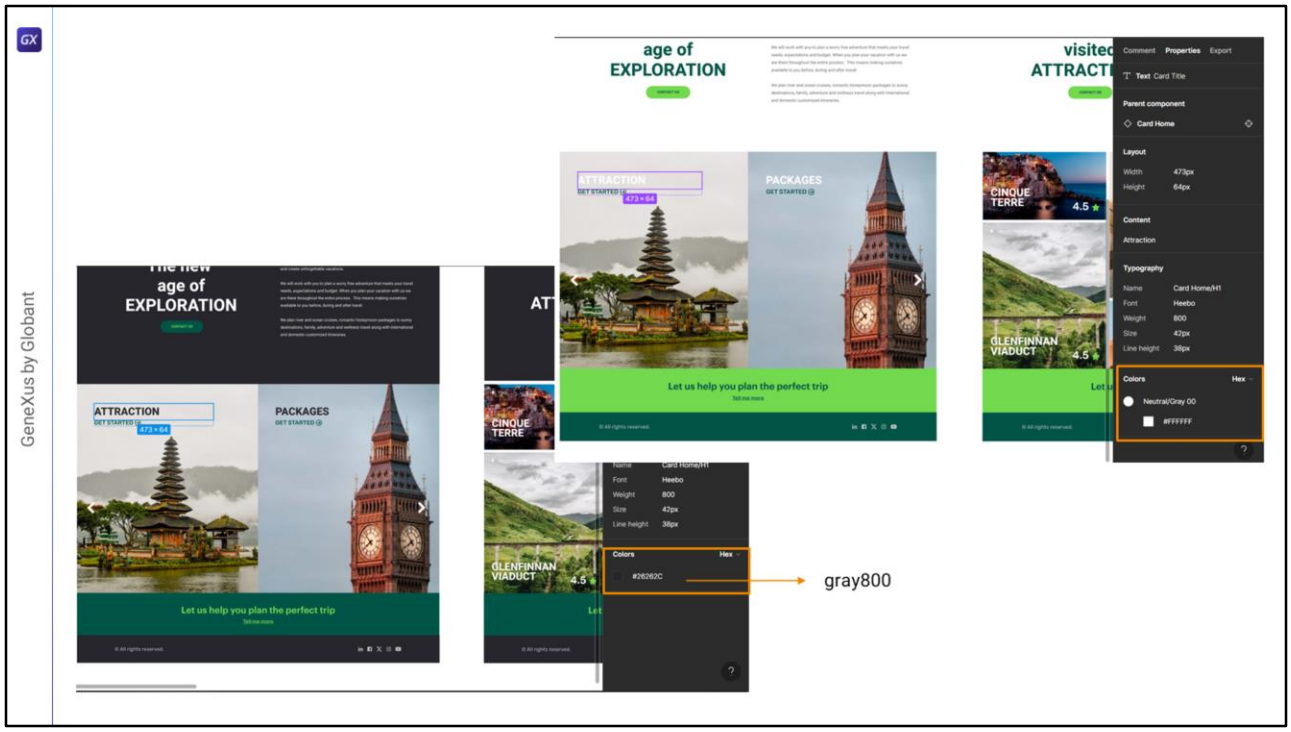


Those things are seen here: surface, title__on-surface, secondary, gray00.

Meanwhile, title__on-image, the first one we had seen, keeps the same white color in both cases.



Now, let's get to what I was asking you, which is what brought up this Dark mode introduction: why had I also isolated a token for the card-home__title? Instead of using, for example, the title__on-image.

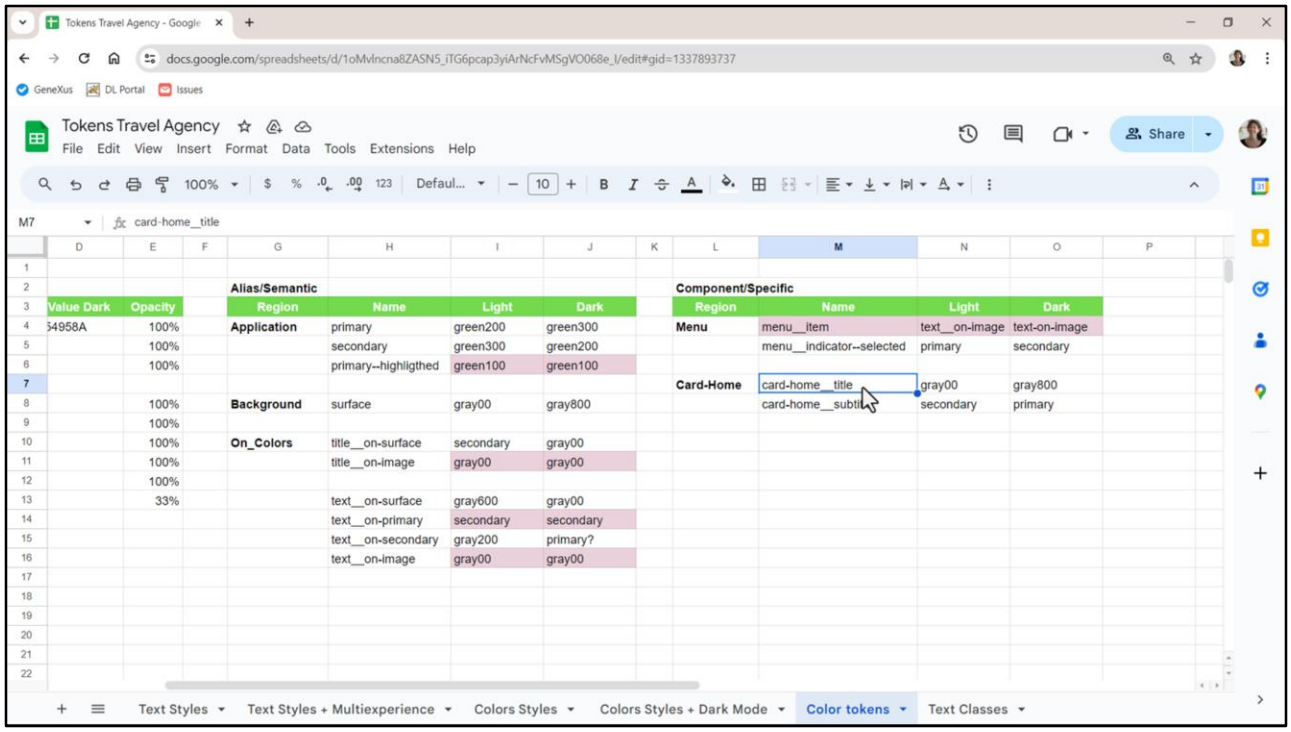


Because while this title__on-image, and even this title__on-image does not change (we see that it is still the same white)...
What about this one?

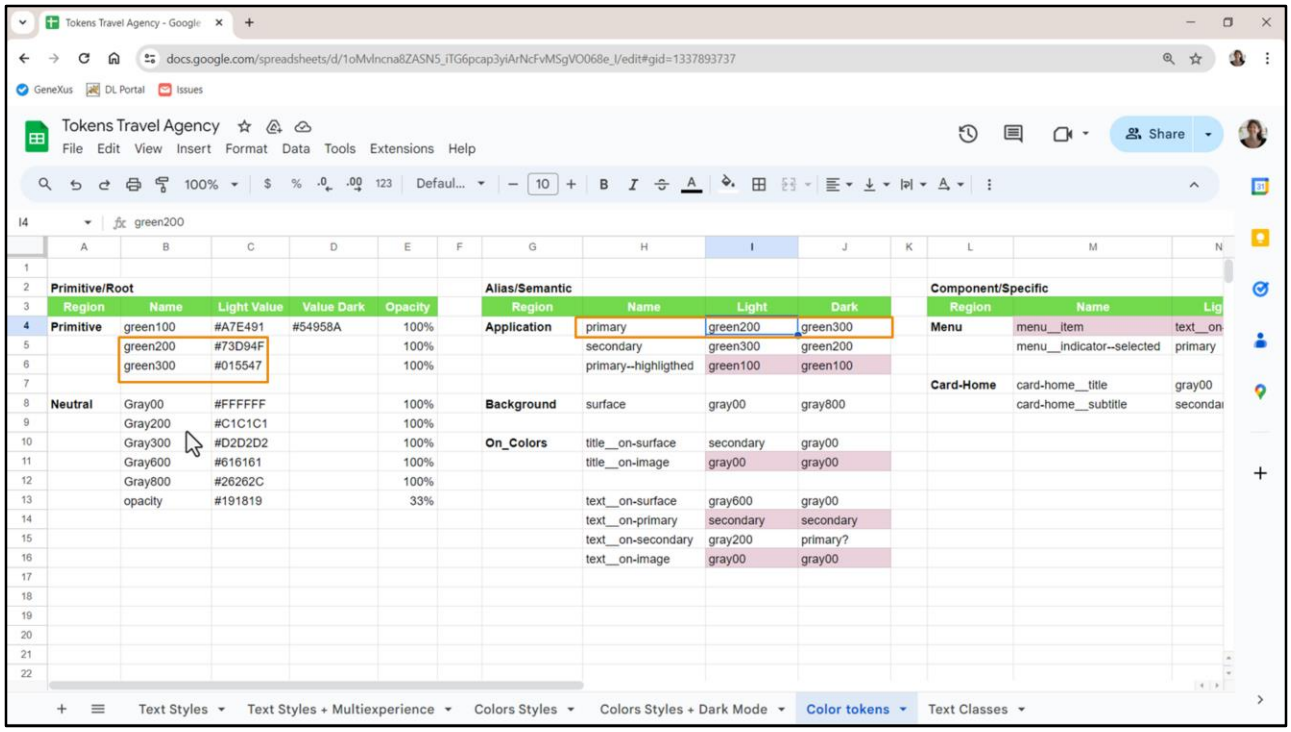
Here we see that it is a sort of dark gray color... well, note that here Chechu is not using the color style. These are inconsistencies that, of course, can happen in any project, right? But clearly, this color is the same as that of the surface... it's the gray800.

OK, but we were saying, here it's gray800, while in the Light mode it's this white.

So we see that this color, the color of this title, behaves differently than the color of the titles on the images used for the generality of the application. It is therefore a particular case. And it is a particular case that only applies to the Home cards, not to anything else, and that's why I made this decision...

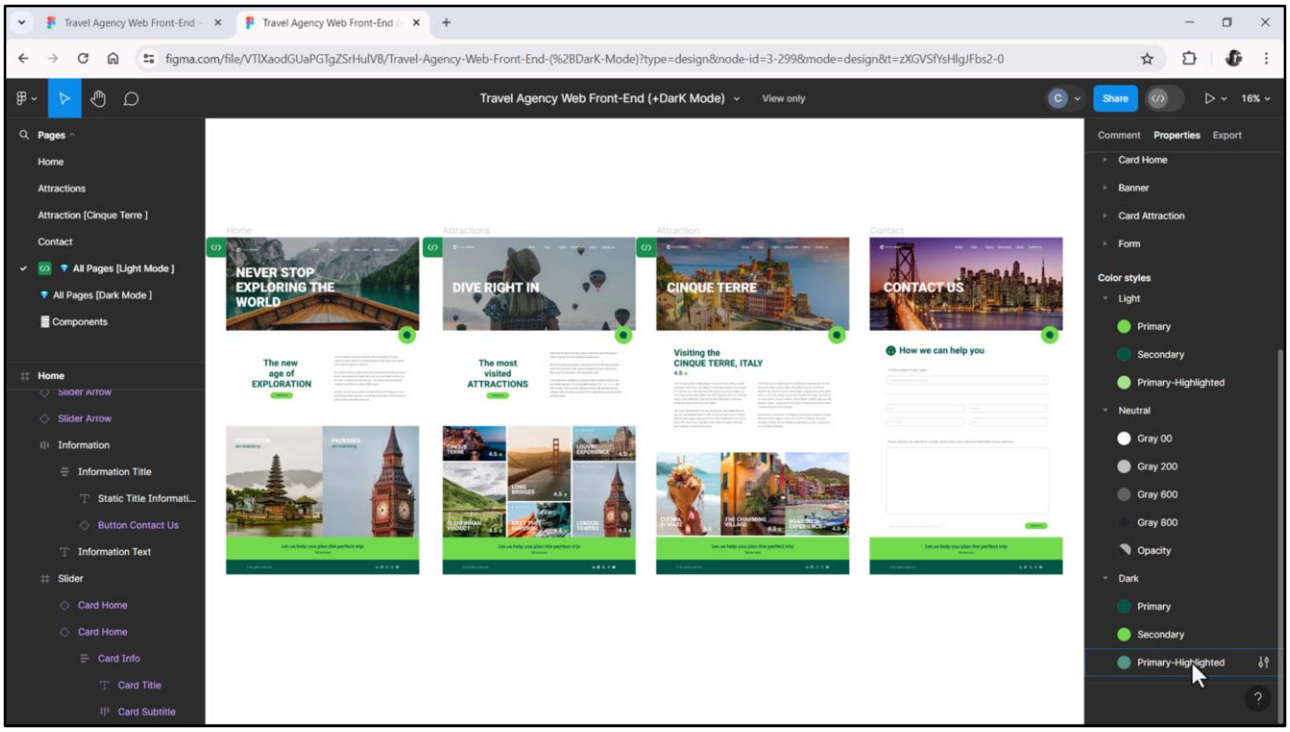


...to define it as a component-specific token.



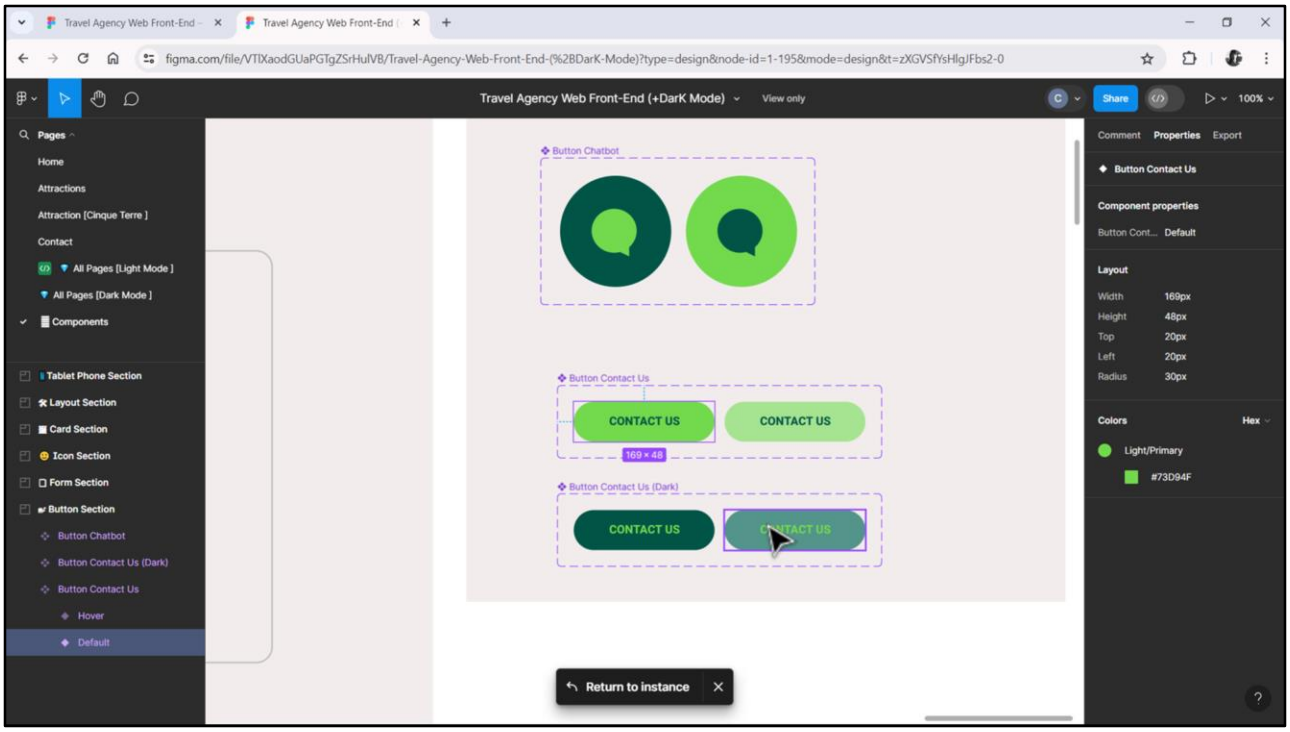
Let's also see what I was saying before, that the primary token is going to be a type of green – green200 – which is the light green, for the Light mode; and the dark green for the Dark mode. And vice versa, they are reversed, for the secondary token.

Here note that the primary--highlighted that the same primary token green100 is using, I called it green100... these do not vary for Light and Dark modes, they will all have the same value for the two modes, except for this first one that I do vary for Light and Dark. And what color does this one correspond to, what color is this?



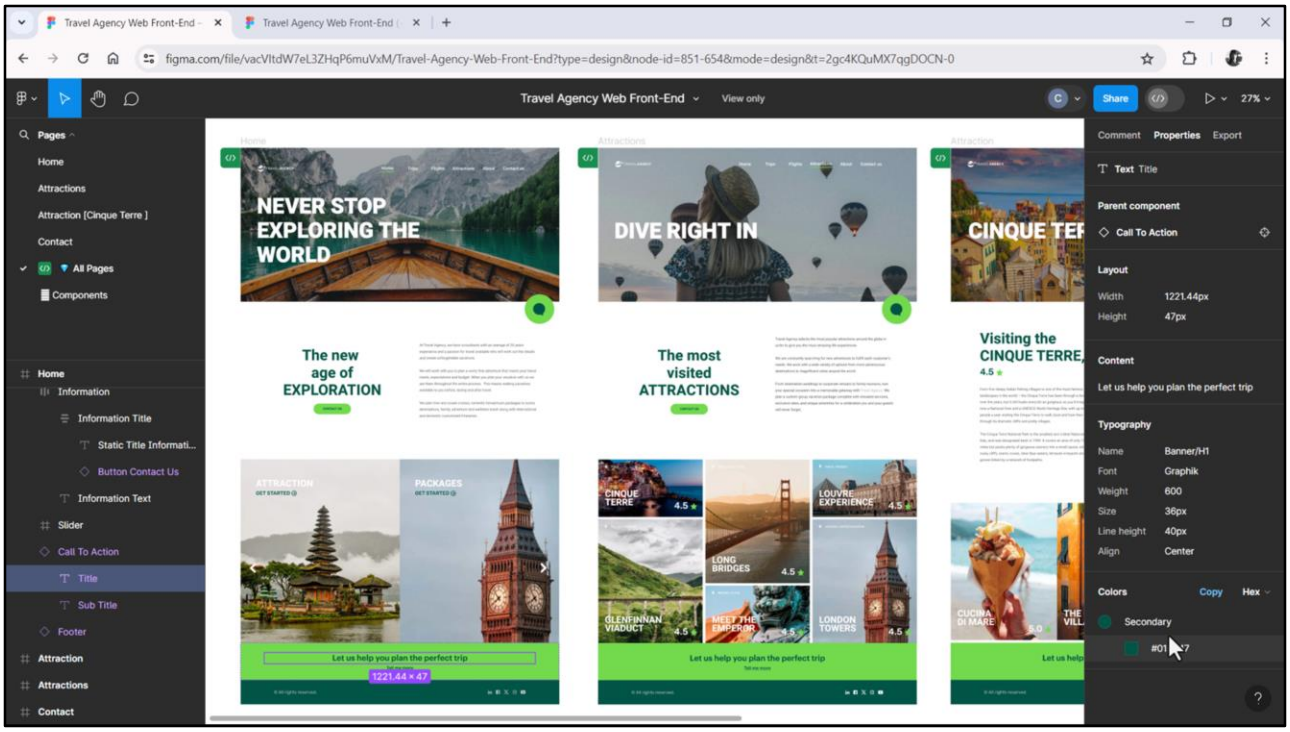
The primary--highlighted one. See that I have this primary--highlighted for the Light mode, and for the Dark mode is this other one, which is lighter than the primary, in both cases.

That was the one that we used, precisely, for the button to...



...let's go to the button to show you... to show you the variations of the button when you hover over it. In the other file we only had this, and in this one we have the two versions: Light and Dark.

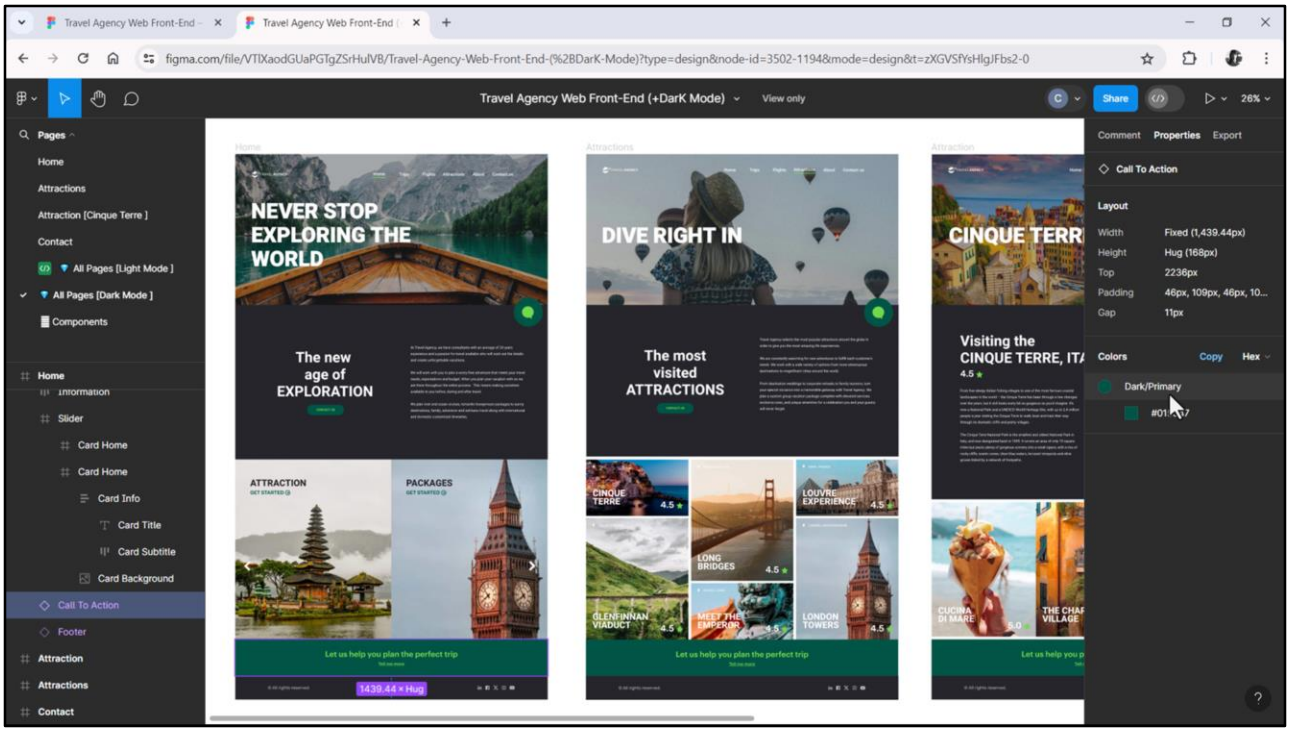
And for the chatbot as well (I had also told you about it in the video where we talked about the images, but there I hadn't presented it yet because I didn't have this file, I didn't want to show you this file that the designer had given me).



To make it easier to compare them, I'll have the two files at the same time. This is the previous one, which has the Light mode only, and this is the one that has both modes. I'm going to select the Dark mode, for an easy comparison.

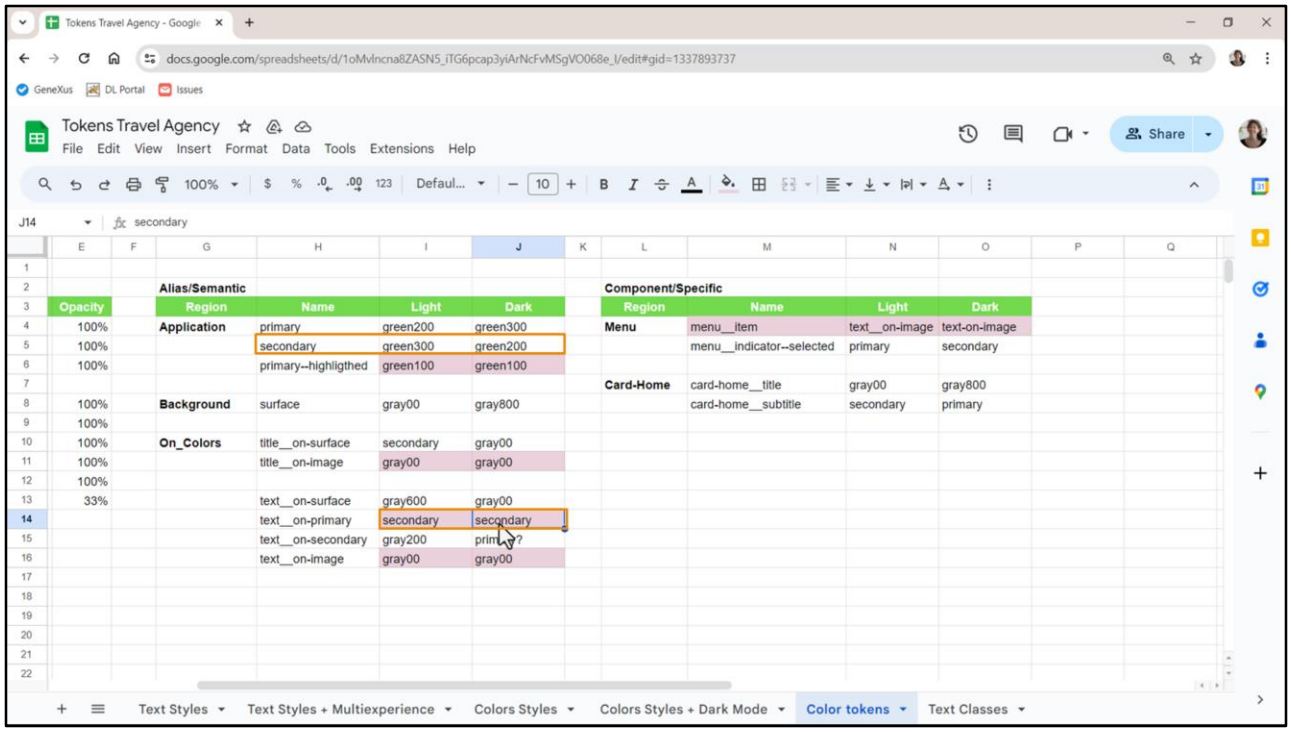
Well, what should we see? That the tokens we have are enough, to see if we don't have to create other tokens, as it happened with the token that we had associated with the card-home for the title.

So, for example, I want to see what happens with primary as background. For the Light mode, we had defined that with primary, with the green one, then we defined a text__on-primary token, which worked for both the button and the banner. Because the color of what was superimposed on that primary color, that is, text__on-primary, was always the same. In this case, in Light mode, we knew that the primary was this green, and the on-primary corresponded to the secondary one.

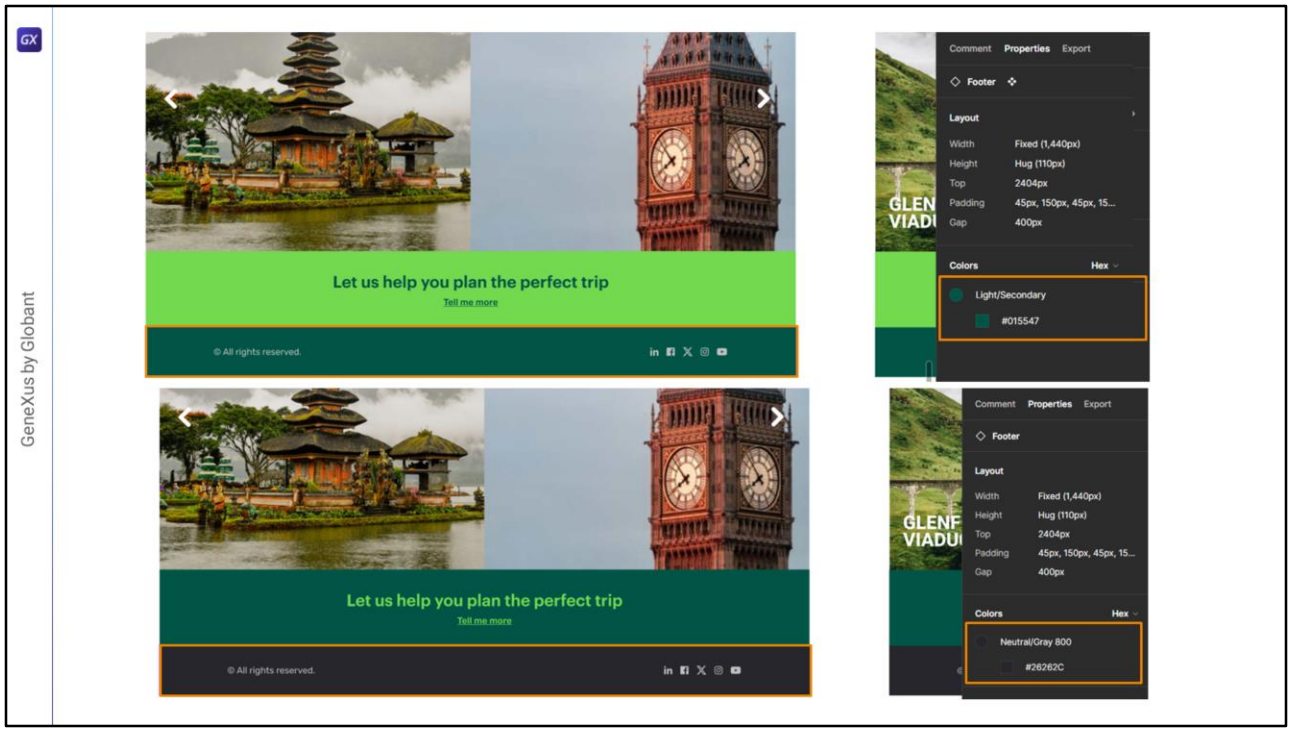


What happens with the Dark mode? Well, we see that they are reversed: what in the other case was primary here is secondary, and vice versa. And that works for us, you see? Because then what do we have to place here?

We know that in this case the primary is this one, and the secondary is the one that in the other case was the primary.



It sounds like a tongue twister but this is where you can understand why we gave it the same value, when in fact it is not the same.
What happens is that this secondary is referring to this secondary token that actually varies between one mode and the other.



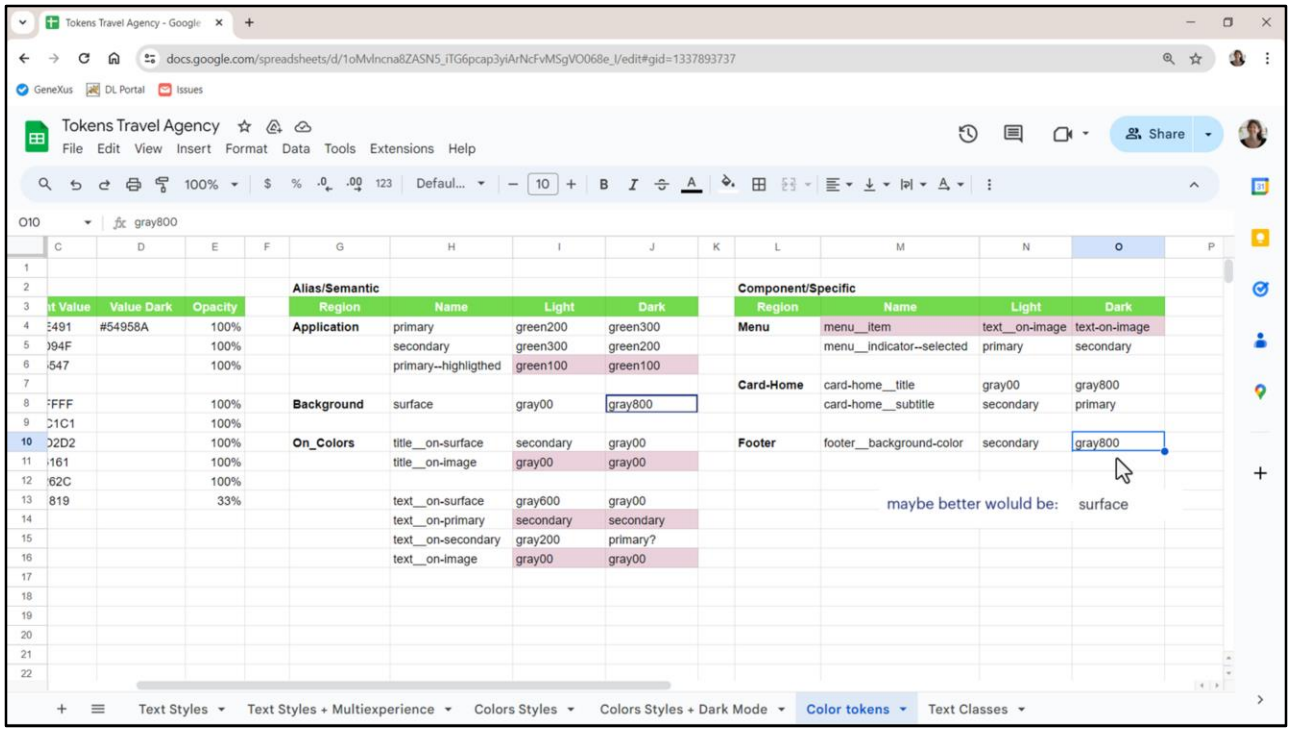
Note that there is an exception to this conversion from primary to secondary and from secondary to primary between Light and Dark modes.

For example, in the chatbot we see that what is primary and secondary is indeed reversed. The same for the button, we had seen, the same for the banner.

But what about the footer? Here that reversion is not being done. The background-color of the footer that is this green for the secondary is not turning into the primary green of the Light mode, that is to say, this light green... if the same rule were followed it would have to be this color. But it is this other one.

So we can clearly see that this is an exception to this rule (which, if followed, should indicate that the secondary here should be the secondary here...) That the secondary here is what? This one. And it's not happening.

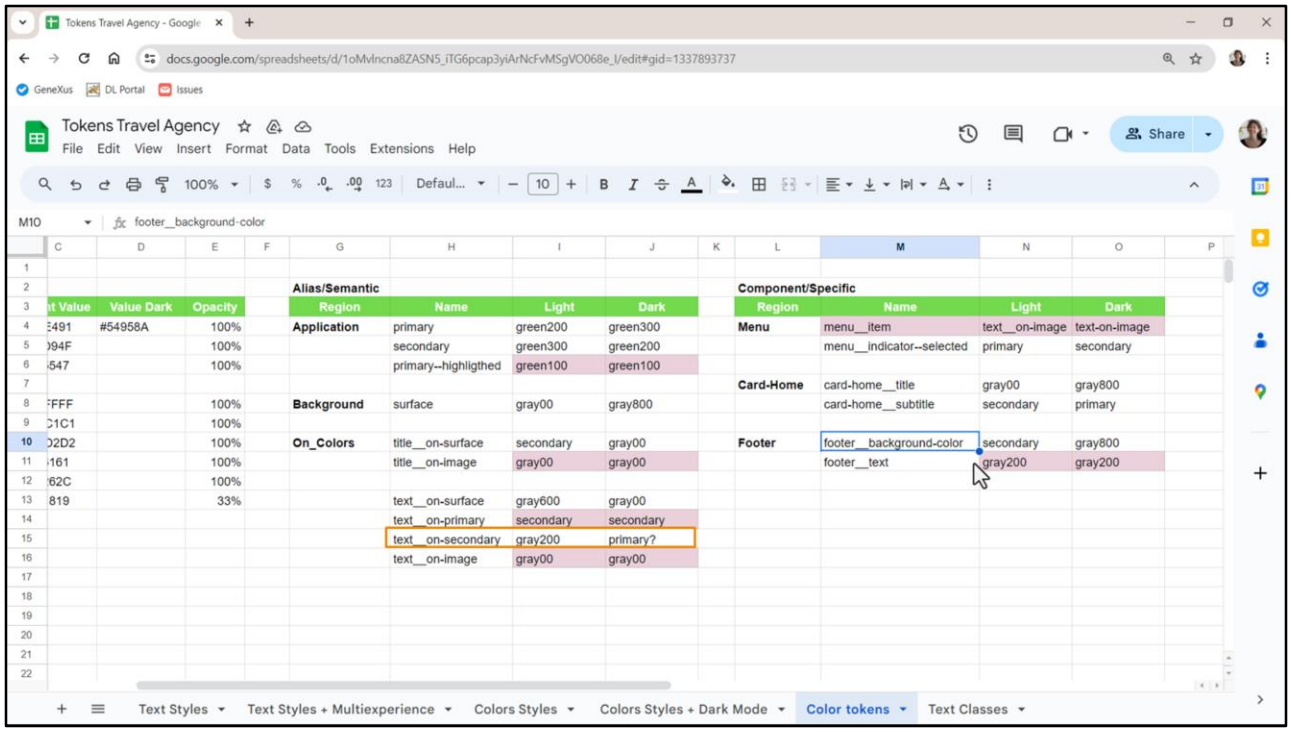
What color is it taking on instead? gray800.



Then I create a special token, footer__background-color, precisely for that exception. It will take the secondary color of the Light mode. And for the Dark mode, it will take this gray800.

And what about the text on that background?

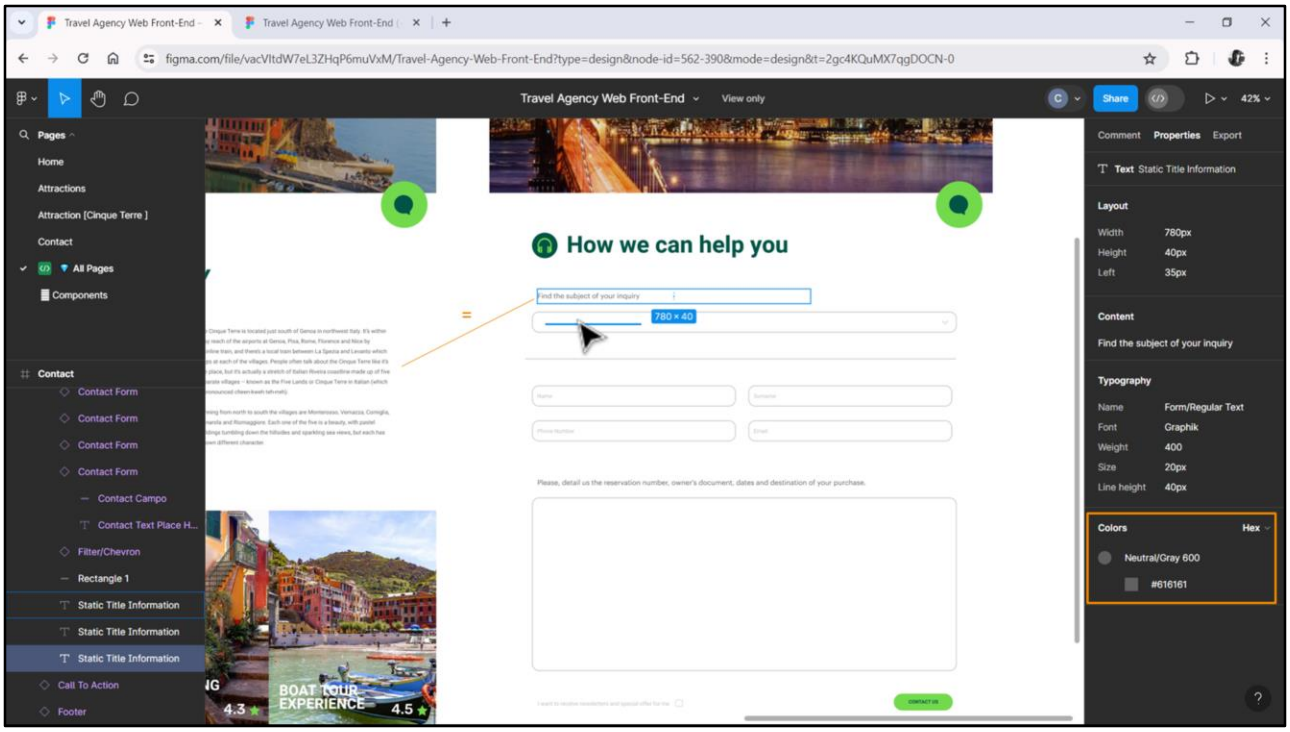
We see that it's this color... if we go to Light mode it's the same one, gray200. Therefore, I add this one as a token, footer__text.



Because it will not correspond to text__on-secondary, because it will not be, for example, the secondary background for the Dark mode; it will be this other one, it will be this background color. So I need to add a special token.

Text__on-secondary ended up like this... this is something to think about because we are not really using it. At first sight, with this color inversion that we were talking about, where the primary of one becomes the secondary of the other mode and vice versa, one could think that actually a text__on-secondary should be primary in both cases if this rule were followed.

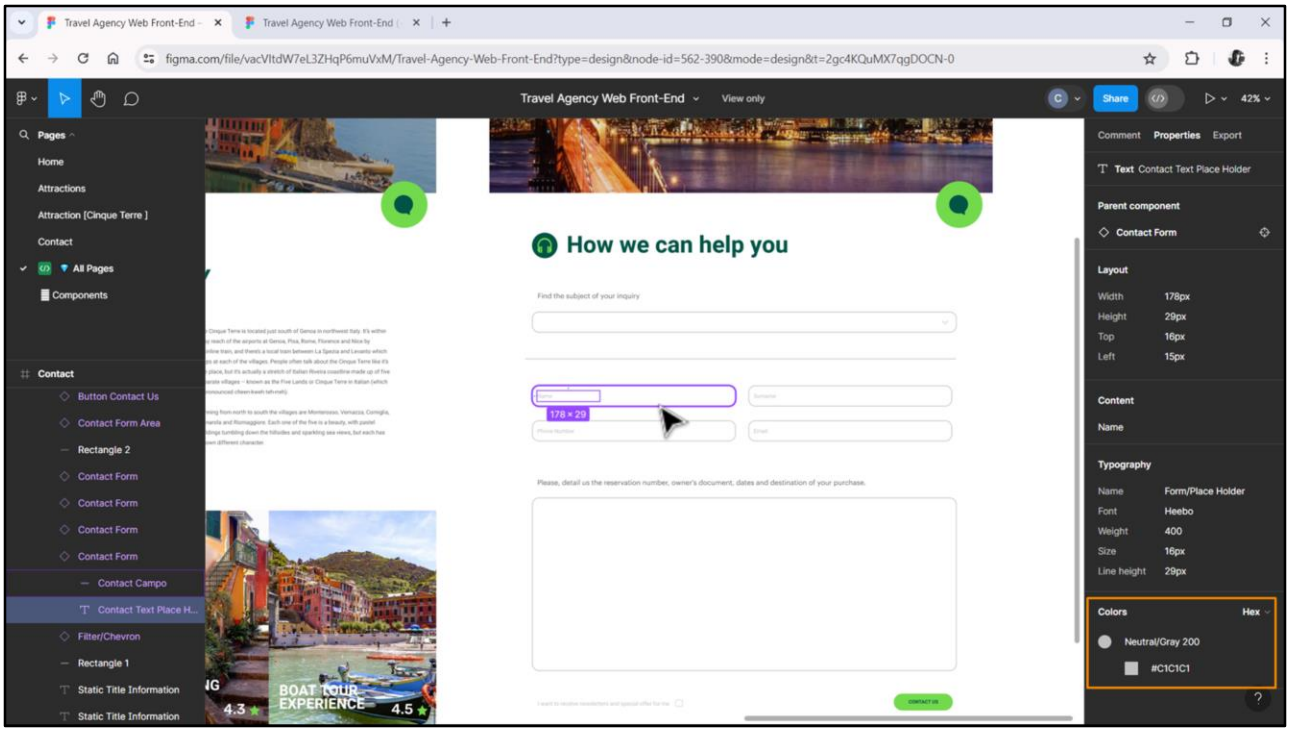
I'm going to leave the two values with a question mark because of what I was saying... at the moment, the only case of secondary as a background is this one, that of the footer. We don't have another case.



To complete the expression of our whole color system, we still have to model only one part of it. Let's look at it here: it has to do with the Contact panel, which is the only panel that has input fields. All the other panels have only output fields.

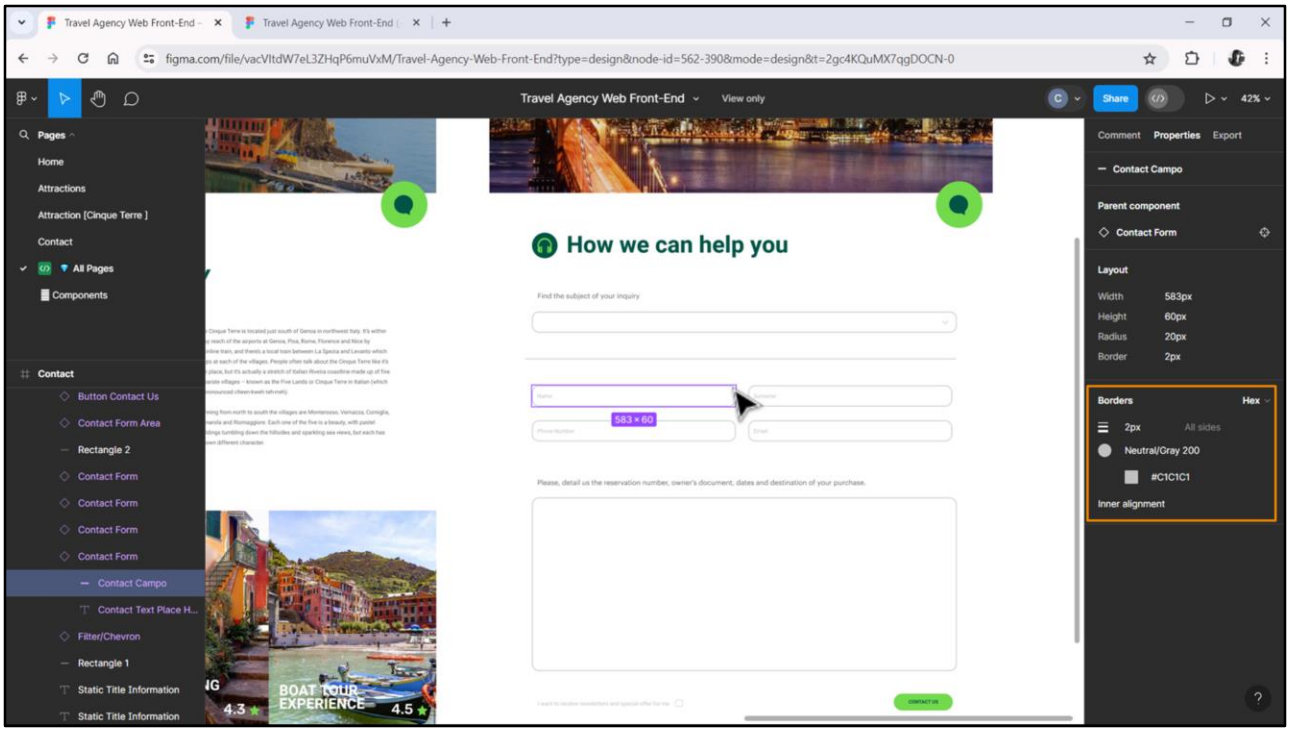
Here the user will have to enter information and press the button.

Note (I'm going to expand it a little more) that this text describing what the user has to enter in the next field takes this color, gray600, which, in this case, is the same as any text on the background surface. So we would set as color that of the same token as this one. There is nothing to specialize there.



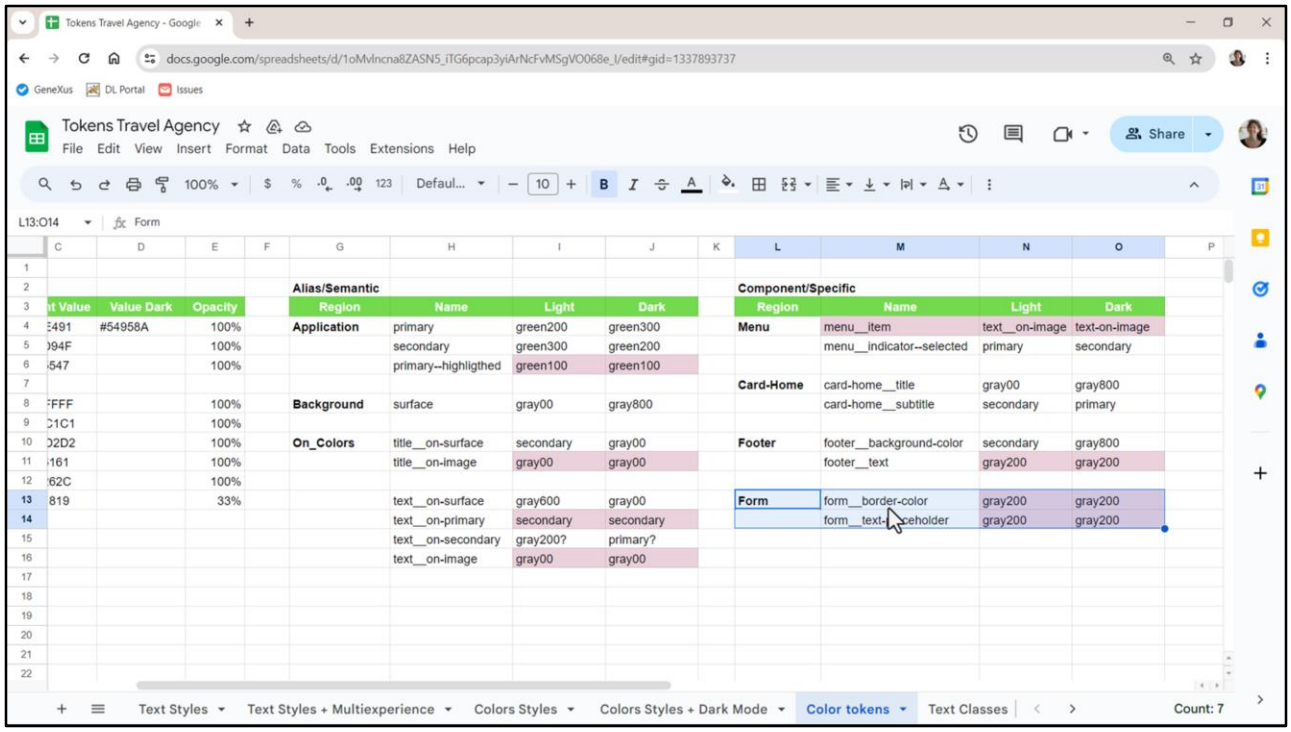
However, note what happens with the text that appears inside here, which is a suggestion text, that is, it is not the description but corresponds to indications for the user. Once the user starts typing, they disappear, unlike this one, which is a description that is always displayed.

This is a lighter color, not gray600. It is gray200, and corresponds to a different concept, which explains the color difference.

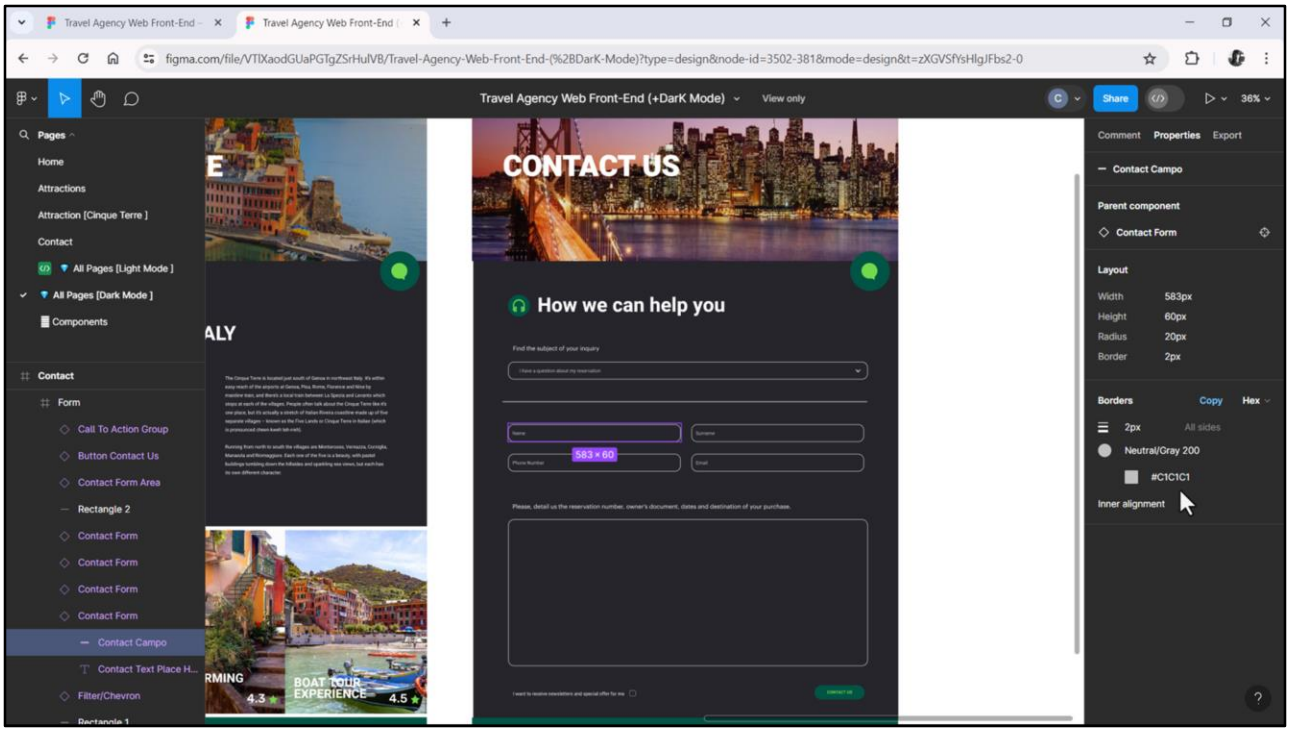


And here is another type of color, that of the borders of these fields. The other fields did not have borders. This one does, and it is gray200 too.

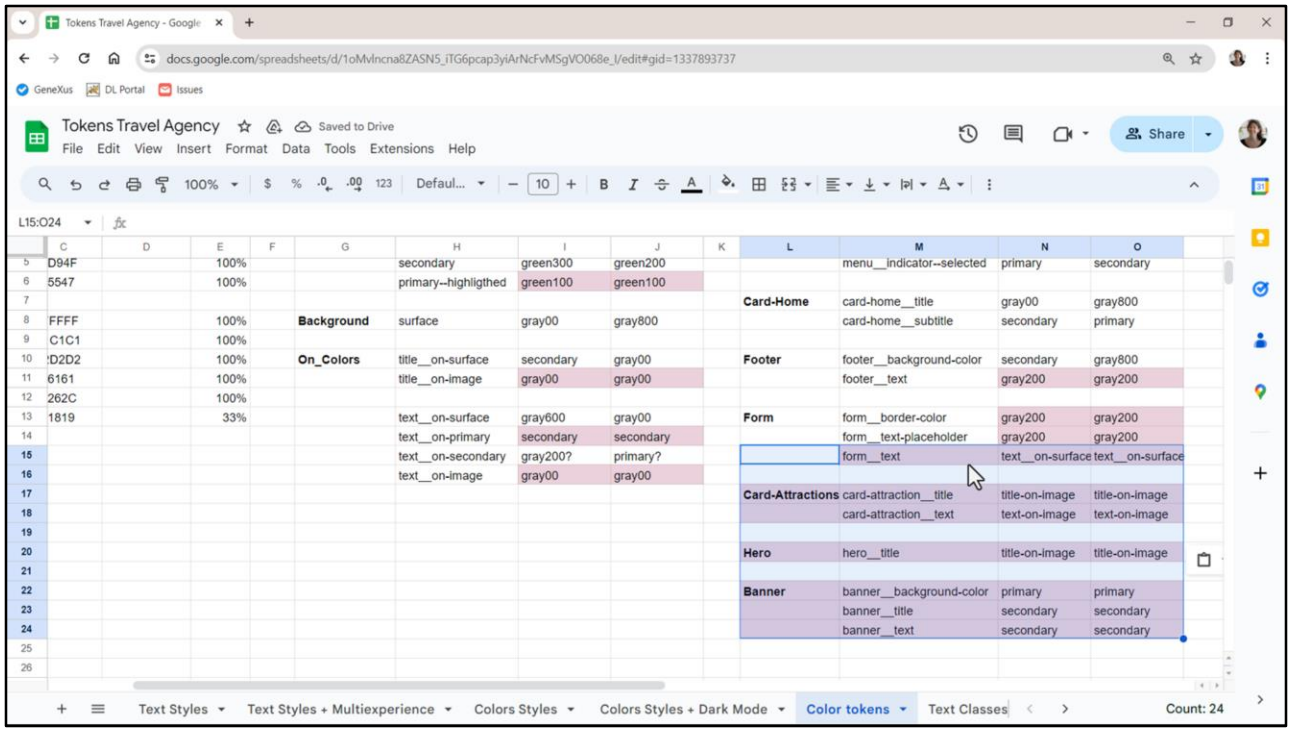
So we still need to model two tokens: one for those borders and another for the text in these cases.



And that's why I build two tokens inside a component that I'm going to name form, because it is specific, very specific. I named one border-color and the other one text-placeholder; it is the text that goes inside the placeholder.



See that it will take for the Dark mode, I didn't show it, the same gray200 value. That's for the border, and for the text inside as well.



With this, we would have the minimum necessary expression of the application's color system.

As I said before, for a system as small as this, we could also create tokens for these other components, which makes it much easier to understand.

For example, we have clearly identified the title that goes on the Hero image, the color, so if we want to change that color, which was using the same title__on-image token, we can change it here directly and separate the title__on-image token from the hero__title; that is, we can start to make them independent. So that, for example, this color can be different from this color here, from this color here, and from this one here.

Well, the same for the others. For the banner, for example, which was not necessary, and for the attraction cards, which were not necessary either.

I marked here everything that in principle would not really be necessary.

And here what is only in these two rows... the ones that repeat exactly the same value. Later we'll see why this might be of interest to us.

Well, we still have to see how to take all this to GeneXus, but as this video is already very long, I'll stop here and continue in the next one.


GeneXus by Globant
GeneXus™
by Globant
training.genexus.com